Completion Delay Minimization for Instantly Decodable Network Coding

by

Sameh Sorour

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy Graduate Department of Electrical and Computer Engineering University of Toronto

Copyright © 2011 by Sameh Sorour
Abstract

**Completion Delay Minimization for Instantly Decodable Network Coding**

Sameh Sorour  
Doctor of Philosophy  
Graduate Department of Electrical and Computer Engineering  
University of Toronto  
2011

Instantly Decodable Network Coding (IDNC) is a subclass of opportunistic network coding that has numerous desirable properties for a wide spectrum of applications, namely its faster decoding delay, simpler coding and decoding processes, and no decoding buffer requirements. Nonetheless, IDNC suffers from two main problems that may limit its attractiveness, as an implementable solution in future wireless networks, against full network coding (FNC), widely studied in the literature. First, it cannot guarantee the decoding of a new packet at each receiver in each transmission, which may severely affect its completion delay. Second, it requires full feedback in order to operate properly, which may be prohibitive for several practical network settings.

In this thesis, we aim to reduce the effect of these drawbacks by studying the problems of minimizing the IDNC completion delay in full and limited feedback scenarios. Since completion delay cannot be optimized only through local decisions in each of the transmissions, we first study the evolution of the IDNC coding opportunities and determine the strategies maximizing them, not only for one transmission, but for all future transmissions. We then formulate the completion delay problem as a stochastic shortest path (SSP) problem, which turns out to be of extremely large dimensions that makes its optimal solution intractable. Nonetheless, we exploit the structure of this SSP and the
evolution of the coding opportunities to design efficient algorithms, which outperform FNC in most multicast scenarios and achieve a near-optimal performance in broadcast scenarios. However, since FNC still outperforms IDNC in some network scenarios, we design an adaptive selection algorithm that efficiently selects, between these two schemes, the one that achieves the smaller completion delay.

To study the effect of feedback reduction, we formulate the completion delay minimization problem, for the cases of intermittent and lossy feedback, as extended SSP and partially observable SSP problems, respectively. We show that these new formulations have the same structure of the original SSP. We thus extend the designed algorithms to operate in intermittent and lossy feedback scenarios, after taking update decisions on the attempted and un-acknowledged packets. These redesigned algorithms are shown to achieve tolerable degradation for relatively low feedback frequencies and high feedback loss rates.
Acknowledgements

First, I would like to express my special thanks to my supervisor, Prof. Shahrokh Valaee, for his continuous academic advice, constant encouragement, guidance and support during my study in his lab at the University of Toronto. I am really grateful to him, not only for contributing many valuable suggestions and improvements to this thesis, but most importantly for training me to be an independent researcher, with the ability to identify interesting and important research problems and to generate frameworks to solve them. Moreover, I am grateful to him for exposing me to collaboration with different industrial and organizational institutions. I feel honored to have been given the opportunity to work under her supervision.

I would also like to express my gratitude to all my professors at Alexandria University, as I owe to them all my knowledge in science and engineering. Without their efforts, I would have never been able to join one of the top institutions in the world, such as the University of Toronto, and to achieve this accomplishment in it.

My appreciation also goes to my colleagues in our research group, for the friendly academic atmosphere, the useful discussions and their encouragement. I am not going to mention names not to forget anyone, but let me express my special thanks to my co-authors, Amin Alamdar Yazdi, Christopher Sue, Ronny Kim and Ju Yong Lee. I would also like to thank my friends in Toronto, for their support in this new country I moved to. Besides, I thank all my friends in Egypt for their loyal friendship, listening ears and kind words that kept me motivated.

Next, I would like to pay my humble respect and the deepest thanks from my heart to my parents. I am extremely grateful to them to have sacrificed themselves to give me the finest education, the warmest care, the righteous upbringing and the best in everything. From my early childhood, they raised me to love learning, and developed my interest in science. I thank them infinitely for their continuous support, patience, encouragement and extra care, which no words can express correctly.
The next warmest thanks from my heart go to my precious wife who really changed my life to the best in everything, and supported me throughout this long student path. She always lightened my bad days and perfected my good ones. No words can express my gratitude and appreciation to her for her kind support and tremendous care, for being my source of inspiration, the twin of my soul and the love of my life.

My final thanks go to my lovely daughter, the beautiful smile in my life, to whom I owe the renewal of my strength and perseverance to accomplish this work.

Sameh Sorour

December 2010.
Contents

1 Introduction ......................................................... 1
   1.1 Preliminary ...................................................... 1
      1.1.1 Multicast Broadcast Services ............................ 1
      1.1.2 Network Coding ............................................. 4
   1.2 Thesis Scope .................................................... 7
      1.2.1 System Model and Parameters ............................ 7
      1.2.2 Problem Statement .......................................... 9
   1.3 Thesis Contributions .......................................... 12
      1.3.1 Contribution 1: Densification of Coding Opportunities in IDNC 12
      1.3.2 Contribution 2: Completion Delay Minimization in IDNC .... 13
      1.3.3 Contribution 3: Adaptive IDNC-FNC Selection ............. 13
      1.3.4 Contribution 4: Extensions to Intermittent and Lossy Feedback 14
   1.4 List of Publications from Thesis ............................ 14

2 Background Work .................................................... 17
   2.1 Important Definitions ........................................ 17
   2.2 Full Network Coding ......................................... 19
   2.3 Opportunistic Network Coding (ONC) ........................ 20
   2.4 Index Coding ................................................... 24
   2.5 Markov Decision Processes (MDP) ............................ 25
4.6.3 Selection Strategy .................................................. 80
4.7 Simulation Results ................................................... 81
4.8 Evolution of the Complete IDNC Graph ......................... 85
4.9 Conclusion ........................................................... 85

5 IDNC Completion Delay .................................................. 87
  5.1 Motivation ........................................................... 87
  5.2 Problem Formulation ................................................ 88
    5.2.1 State Space S .................................................. 88
    5.2.2 Action Spaces A(s) ............................................ 89
    5.2.3 State-Action Transition Probabilities ....................... 89
    5.2.4 State-Action Costs ............................................ 90
    5.2.5 SSP Solution Complexity ..................................... 91
  5.3 Properties of Formulated SSP ..................................... 91
    5.3.1 SSP Structure .................................................. 91
    5.3.2 Geometric Structure .......................................... 94
    5.3.3 Evolution of Action Space .................................... 98
  5.4 Proposed Algorithms ............................................... 100
    5.4.1 Maximum Weight Clique Selection Algorithm ............... 100
    5.4.2 Maximum Weight Vertex Search Algorithm .................. 101
    5.4.3 Complexity of the Heuristic Algorithm .................... 103
  5.5 Simulation Results ............................................... 104
  5.6 Conclusion ........................................................ 114

6 Adaptive IDNC-FNC Selection .......................................... 116
  6.1 Motivation ........................................................ 116
  6.2 FNC Completion Delay Metrics .................................. 117
  6.3 IDNC Completion Delay Metrics .................................. 118
6.3.1 Theoretical Foundation ........................................ 119
6.3.2 Approach I ...................................................... 120
6.3.3 Approach II ....................................................... 123
6.3.4 Approach III ....................................................... 127
6.4 Adaptive Network Coding (ANC) Scheme ....................... 131
6.5 Simulation Results .................................................. 133
   6.5.1 Study of the o(1) Term ...................................... 133
   6.5.2 Performance Testing .......................................... 138
6.6 Conclusion .......................................................... 142

7 IDNC with Intermittent Feedback .................................... 144
   7.1 Motivation .......................................................... 144
   7.2 Problem Formulation .............................................. 146
      7.2.1 State Space $S$ ............................................... 146
      7.2.2 Action Spaces $A(s)$ ....................................... 147
      7.2.3 State-Action Transition Probabilities .................... 147
      7.2.4 State-Action Costs ......................................... 148
      7.2.5 SSP Solution Complexity .................................. 148
   7.3 Properties of the SSP Formulation ............................. 148
   7.4 Vertex and Graph Update Approaches .......................... 150
      7.4.1 Full Vertex Elimination (FVE) ............................ 150
      7.4.2 Full Vertex Elimination with Graph Update (FVE-GU) .... 151
      7.4.3 Stochastic Vertex Elimination (SVE) ....................... 151
      7.4.4 Stochastic Vertex Elimination with Graph Update (SVE-GU) 152
   7.5 Proposed Algorithm ............................................. 152
   7.6 Simulation Results ............................................... 153
   7.7 Non-Periodic and Asynchronous Intermittent Feedback .......... 162
   7.8 Conclusion ........................................................ 163

ix
8 IDNC with Lossy Feedback

8.1 Motivation .................................................. 164
8.2 Problem Formulation ........................................... 165
  8.2.1 Observation Spaces $\mathcal{O}(\kappa)$ ................. 166
  8.2.2 Conditional observation probabilities $Q_\kappa(s',o)$ .... 166
  8.2.3 Belief States and Belief Updates .................... 167
8.3 Formulation Properties and Proposed Algorithms ........... 169
8.4 Simulation Results ............................................ 171
8.5 Conclusion .................................................. 178

9 Conclusion and Future Directions ................................. 179

9.1 Thesis Conclusion ............................................. 179
  9.1.1 Densification of Coding Opportunities in IDNC .......... 180
  9.1.2 Completion Delay Minimization in IDNC ............... 180
  9.1.3 Adaptive IDNC-FNC Selection .......................... 181
  9.1.4 Completion Delay Minimization with Intermittent Feedback ... 182
  9.1.5 Completion Delay Minimization with Lossy Feedback ...... 183
9.2 Future Directions .............................................. 184
  9.2.1 Network Topology ...................................... 184
  9.2.2 Channel Model .......................................... 185
  9.2.3 Transmission Mode ..................................... 186

Bibliography .......................................................... 188
List of Tables

4.1 Evolution Cases for Each Pair of Receivers ........................................ 55
4.2 Evolution Cases for Each Pair of Receivers in the broadcast scenario . . . 64
List of Figures

1.1 System model and packet status after the initial transmission phase . . . 7

3.1 Example of a feedback matrix and its G-IDNC graph . . . . . . . . . . 38
3.2 The S-IDNC graph for the example of Figure 3.1 . . . . . . . . . . . . . . 41

4.1 Pairwise Edges between receivers $i$ and $k$. . . . . . . . . . . . 51
4.2 Illustration of Explanation 2. . . . . . . . . . . . . . . . . . . . . . 57
4.3 Illustration of Explanation 3. . . . . . . . . . . . . . . . . . . . . . 58
4.4 Illustration of Explanation 5. . . . . . . . . . . . . . . . . . . . . . 59
4.5 Illustration of Explanation 9. . . . . . . . . . . . . . . . . . . . . . 62
4.6 Illustration of Explanation 10. . . . . . . . . . . . . . . . . . . . . . 63
4.7 Average coding density evolution for $M = 60$ and $N = 30$ . . . . . . 83

5.1 SSP state for the example in Figure 3.1 and potential transitions for $a_7$ . 90
5.2 Geometric structure of the example in Figure 5.1 without packet 4 . . . . 95
5.3 Average completion delay of different schemes and norms vs $\mu$ . . . . . 105
5.4 Average completion delay of different schemes and norms vs $M$ . . . . . 106
5.5 Average completion delay of different schemes and norms vs $N$ . . . . . 107
5.6 Average completion delay of different schemes and norms vs $p$ . . . . . 108
5.7 Average completion delay of opt and srh algorithms vs $\mu$ . . . . . . . 110
5.8 Average completion delay of opt and srh algorithms vs $M$ . . . . . . . 111
5.9 Average completion delay of opt and srh algorithms vs $N$ . . . . . . . 112
Chapter 1

Introduction

1.1 Preliminary

1.1.1 Multicast Broadcast Services

Multicast Broadcast Services (MBS) have become a cornerstone in the design of all future wireless standards and networks, such as LTE [2], WiMAX [1] and satellite networks. This growth of interest in MBS is a natural response to both the proliferation of applications, which can be simultaneously requested by multiple co-located receivers, and the expectations of these receivers to simultaneously access different subsets of such applications. Examples of these applications are internet TV, video news feeds, file downloads, and location based applications such as location based advertisements and queries for location based services.

To give an example of such scenarios, imagine a group of receivers, in the coverage area of a wireless sender, all requesting to watch a soccer match on their smart phones, similar to the service offered by the CBC channel during the 2010 world cup. In this case, all of these receivers need to correctly receive all the packets of this match stream, which we refer to as a broadcast scenario. Now assume that some of these receivers are also downloading files, others are watching the breaking news and others are looking for the
closest restaurants or using voice over IP applications,... etc. In this case, each of these receivers will be receiving the match stream packets as well as other packets that may or may not be common with some subsets of other receivers. We refer to these scenarios as multicast scenarios.

Due to the high demand on MBS applications and their high bandwidth requirements, it is very important for MBS protocols to efficiently utilize the scarce bandwidth resources available to the network, by improving the system bandwidth efficiency. In other words, the MBS protocols should minimize the amount of resources required to efficiently deliver each group of packets, in order to increase the transmission rate of new packets (i.e. throughput). At the same time, these protocols should be able to guarantee the quality of the streaming applications, in which the received packets should be always useful at their reception instant, in order to prevent interruption or flickering of the stream. In other words, the streaming packets should not experience high delays until their correct reception and passing to upper layers.

Other than cellular and WiMAX systems, the concept of MBS can be also very important in networks of sensors and/or robots (we will refer to both as agents), having processing and command execution capabilities [59]. These networks generally consists of a fusion or command center and numerous agents scattered in a region, such as a factory, an oil field, a mine or a battlefield. Each agent in this network should be able to receive, process and execute one or more complex commands in the mission it is handling. Each command and its associated data is dispatched from the center in one or several packets. In a lot of situations, an agent may need to coordinate with a subset or all the other agents while executing its commands, in order to achieve the best outcome or to avoid any accidents due to the lack of organization. Thus, each agent needs to know its own commands as well as the commands of a subset or all the other agents. Therefore, the dissemination of these command packets to the agents can be viewed as multicast or broadcast scenarios. In these applications, in-order acquiring and processing of the
commands may not be a real issue. However, fast command execution may be crucial and therefore, it is imperative that new packets arrive at the destinations as quickly as possible regardless of their order [59]. Moreover, all coordinating agents need to learn about each other’s commands as fast as possible to avoid wrong actions.

In all the above scenarios, the fact of MBS packet transmission over wireless channels threatens the simultaneous achievement of the throughput and delay goals, because these packets are subject to loss due to the severe impairments of the wireless channel, such as fading, shadowing, multipath and interference. These losses are generally perceived at higher layers as packet erasures. In order to deliver these lost packets to their intended receivers, efficient packet recovery protocols should be carefully designed, so that they do not greatly degrade the MBS throughput and delay performances in multicast and broadcast scenarios.

Most employed packet recovery protocols in current network standards can be looked at as different variants and tailoring of the well-known automatic repeat request (ARQ) protocols, and their combinations with forward error correction (FEC) schemes, known as the hybrid automatic repeat request (HARQ) protocols. Both protocol families recover lost packets by retransmitting them separately, which considerably limits the number of packets recovered per retransmission and the number of receivers benefiting from each of them. In other words, the sender may have to rebroadcast a lost packet to all the receivers, although there may be only one receiver that did not correctly receive that packet. This inefficiency results in longer packet recovery phases in order to satisfy all receivers’ demands, which violates the desired levels of throughput and delay in MBS applications. Obviously, the larger the MBS receiver pool, the greater and more severe the effect of this packet recovery inefficiency. These limitations called for a new approach to increase the efficiency of the packet recovery process. One major breakthrough in this area came with the development of network coding.
1.1.2 Network Coding

Since its first introduction in [3], network coding has been a great attraction to numerous studies, as a routing and scheduling scheme that attains maximum information flow in a network. Different studies investigated its ability to achieve this maximum information flow in different network topologies and transmission modes (multicast, unicast). The main core of network coding is the idea of packet mixing at intermediate network nodes in order to achieve the maximum information usefulness of each coded packet. Several works have studied the generation of efficient network codes for both acyclic [39] and cyclic networks [17, 27].

After its strong establishment in wired and ad-hoc network settings [24, 30–32, 34, 35, 41, 53], the idea of employing network coding as a packet recovery scheme in centralized one-hop networks was introduced in several works for unicast, multicast and broadcast scenarios [16, 18, 19, 28, 29, 33, 40, 48–52, 58–62, 73–77, 79]. This approach was referred to as the Multiuser ARQ in [40]. For wireless unicast and multicast scenarios, network coding exploits the natural properties of overhearing and packet reception diversity over wireless broadcast channels. By overhearing, we mean the event of receivers detecting packets that are not requested by them. These two properties can be exploited to increase the information content of each recovery packet, so as to serve a larger number of receivers.

To give an example, consider two receivers A and B that request packets 1 and 2, respectively, from the sender. After an initial transmission of packets 1 and 2, assume that A correctly received packet 2 only and B correctly received packet 1 only. Consequently, both receivers did not get their requested packets but overheard each other’s packets after their initial transmission from the sender. In conventional ARQ schemes, the sender must re-send packets 1 and 2 in two separate transmissions. If network coding is allowed, both A and B can get their requested packets in one recovery transmission combining packets 1 and 2. This reduces the number of transmissions to recover these two packet by 50% compared to conventional ARQ. Even if both A and B requested both packets 1 and
2, the diversity in their reception of packets (A has 2 and B has 1), after the initial transmission, allows the mixing of these packets to deliver the other requested packet to each of them more efficiently, thus achieving the same performance improvement. In all unicast, multicast and broadcast scenarios, network coding has shown great abilities to substantially improve the packet recovery performance.

Among different works on network coding for efficient packet delivery over erasure channels, two network coding approaches can be distinguished in the literature, namely full [24] [53] and opportunistic [31] [30] network coding. In the full network coding (FNC) approach, all the packets of the frame are combined with deterministic or random non-zero coefficients in either all the transmissions or the recovery transmissions only. Consequently, FNC aims to deliver all the packets of the frame to all receivers. In [18] and [19], FNC was proved to achieve optimal throughput in broadcast scenarios. However, this optimality is achieved at the cost of large delays in packet decoding, since the receivers need to collect a large number of coded packets before being able to decode all of them simultaneously. Obviously, this may be proper for delay insensitive applications, but is definitely not suitable in applications that does not tolerate such packet delays, such as real-time and command dissemination applications. Moreover, FNC is throughput inefficient in multicast scenarios since it must deliver all the packets to all receivers, including the non-requested ones, which usually requires more transmissions than delivering the requested packets only.

On the other hand, the opportunistic (or feedback-based) network coding (ONC) approach exploits the knowledge of received packets, at the different receivers, in strategically selecting the recovery packet combinations, in order to guarantee the decodability of new packets after each recovery transmission for a subset of or hopefully all the receivers. Obviously, this network coding approach is much more suitable for ensuring the quality of real-time and command dissemination applications, since decoded packets can be passed directly to the upper-layers and employed at the destination. For ordinary
Chapter 1. Introduction

real-time applications, [73, 74] proposed opportunistic schemes that deal with in-order packet decoding and delivery at the receivers. A second set of opportunistic schemes were designed in [14, 33] for applications, in which decoded packets should bring new information to the receivers, irrespective of their order. These schemes are more suitable for real-time streams encoded using multiple description source coding, coordinated command dissemination to sensors/robots and location based information [59].

[59, 60] considered a subclass of this ONC approach, in which the received coded packets at any receiver are allowed to be decoded only at their reception instant, and cannot be stored for future decoding. We refer to this subclass as the instantly decodable network coding (IDNC). The interest in this subclass can be justified by its numerous desirable properties such as fast and progressive packet decoding, simple encoding and decoding procedures and no buffer requirements for the network coding layer [59].

The order-insensitive ONC and IDNC approaches are definitely more efficient than the order-constrained approaches in achieving higher throughput, since they do not have any order restrictions on the selection of packets to be encoded in each transmission. We refer to the packets encoded for a transmission as its packet combinations. They can thus generate more efficient packet recovery combinations, which allows them to reduce the number of recovery transmissions and thus complete the packet recovery phase earlier. This property enables the transmission of new packets faster and thus increases the system throughput. However, the length of the packet recovery phase in these approaches greatly depends on their strategy of selecting and scheduling the packet combinations. In other words, two different scheduling policies of packet combinations may end up with totally different recovery completion times, and thus different throughput.

The problem of minimizing the number of opportunistic network-coded recovery transmissions has been considered for error-free channels and has been proved to be NP-hard [16] and hard to approximate [38]. Nonetheless, the problem of minimizing the same parameter over erasure channels is still open for research, and thus will be the scope
Chapter 1. Introduction

1.2 Thesis Scope

1.2.1 System Model and Parameters

The system model we consider in this thesis is depicted in Figure 1.1. It consists of a wireless sender that is required to deliver a frame (denoted by $\mathcal{N}$) of $N$ source packets to a set (denoted by $\mathcal{M}$) of $M$ receivers. Each receiver is interested in receiving either a subset or all the packets of $\mathcal{N}$. The former case is referred to as multicast whereas the latter case is referred to as broadcast. We will refer to the requested and undesired packets of any receiver by its primary and secondary packets. Each receiver $i$ is subject to independent packet erasures (successful reception) with probability $p_i$ ($q_i = 1 - p_i$), which is assumed to be fixed during the frame transmission period. This channel model is widely used in almost all similar network coding works [6,14,18,19,33,40,47–52,59–61,73–76,79] and is suitable for slow fading channel scenarios, such as stationary and pedestrian users in
cellular networks, as well as wireless sensors and robots.

The sender initially transmits the $N$ packets of the frame “uncoded” in an *initial transmission phase*. By uncoded, we mean that these packets are not combined with other packets in a network coding fashion. Nonetheless, they may be channel-coded within themselves to detect and/or correct errors in them. Each receiver listens to all transmitted packets (even the ones that it does not want) and feeds back to the sender a positive acknowledgement ACK for each received packet. After the initial transmission phase, a recovery transmission phase starts to provide the receivers with their missing requested packets. In this recovery phase, the sender exploits the diversity of received, lost and overheard packets at different receivers to generate IDNC packets, which are instantly decodable for a large number of receivers. To reduce the feedback load, only the receivers, which correctly decode a new source packet from a transmission, send an ACK for this transmission. Indeed, an IDNC packet is useless for receivers that cannot decode it, and thus their feedback of receiving or not receiving the packet is not needed at the sender. Knowing the number of receivers that can decode a packet in advance, the sender allocates time to listen to feedback from these receivers only.

Let $\mu_i$ be the demand ratio of receiver $i$, defined as the ratio of the requested packets of $i$ in the frame to the total frame size $N$. In this case, we can consider the broadcast scenarios as a special case of the corresponding multicast scenarios, in which the demand ratios for all receivers are equal to one. Also, define $\mu$ as the average of the demand ratios of all receivers, which is equal to:

$$\mu = \frac{1}{M} \sum_{i=1}^{M} \mu_i . \tag{1.1}$$

In the rest of the thesis, we say a receiver is targeted by a coded packet or a packet is targeting a receiver if this packet is instantly decodable by this receiver.

We define the system throughput (also called bandwidth efficiency) as the expected
fraction of the bandwidth employed in transmitting original (i.e. non recovery) packets from the sender. According to the above model description, this quantity can be expressed as the ratio of the frame size to the expected number of initial and recovery transmissions until all receivers obtain their requested packets. Thus, the system throughput $\eta$ can be expressed as:

$$\eta = \frac{N}{N + E[D]}$$  \hfill (1.2)

where $D$ represents the number of recovery transmissions needed to complete the delivery of all requested packets to their intended receivers. In other word, this term is equal to the number of transmissions from the start of the packet recovery phase until the last receiver completes its reception to its last requested packet. We will refer to this term as completion delay. It is obvious from (1.2) that maximizing the system throughput is equivalent to minimizing the expected completion delay.

### 1.2.2 Problem Statement

In this study, we are only interested in the IDNC subclass of network coding, due to its very attractive and desirable properties. Nonetheless, being a subclass of ONC, IDNC suffers from two major problems:

1. IDNC cannot guarantee the decoding of a new packet by each receiver in each transmission.

2. IDNC requires the reception of feedback from each of the receivers after each transmission in order to decide on which packets to encode in the subsequent transmissions.

The first problem has different implications on the ability of the sender to control the expected completion delay and the system throughput of IDNC. Indeed, the completion delay of IDNC can be highly variable according to the policy with which the sender
schedules packet combinations for IDNC transmissions. This problem becomes more serious in scenarios with large numbers of receivers, in which FNC may achieve a better completion delay performance than IDNC. It is also proved that FNC can achieve the optimal completion delay performance in broadcast scenarios [47].

These implications about the IDNC completion delay raise two main concerns. The first concern is how to optimize the scheduling of IDNC packets in order to minimize the average completion delay. One can intuitively assume that the answer to this question is to simply serve the maximum number of receivers in each transmission, as long as the instant decodability property is maintained. However, this simple intuition may not be true and a deeper investigation of the problem is required. The second concern is how to adaptively switch between IDNC and FNC in applications where minimizing the completion delay cannot be compromised.

The second problem may represent a severe limitation in the practicality of IDNC (and ONC in general) since the amount of feedback needed for IDNC to function properly may be prohibitive in some applications and network settings. Another problem is that, in many network settings, a feedback cannot be received before several packet transmissions. One important example is the time division duplex based networks, in which several packets can fit into each downlink frame. A last problem is the fact that reverse channels in many networks are also lossy and thus part of the feedback will reach the sender. These scenarios raise the concern of how to deal with this lack of information and feedback, when scheduling IDNC transmissions, in order to maintain the performance degradation in the completion delay and system throughput as low as possible.

Another important concern in IDNC is the effect of each IDNC transmission on the remaining coding opportunities for the subsequent transmissions. Although a transmission may be selected to optimize a desired parameter for this particular transmission, this same parameter may be less optimized in subsequent transmissions due to a resulting lower number of coding opportunities. On the other hand, a sub-optimal solution at one
or several transmissions may result in much diverse coding opportunities in the following transmissions, which may result in an overall better performance. This effect is more dominant in parameters that are not determined in each transmission but at the end of the frame transmission, such as the completion delay. Since completion delay is the parameter of interest in this thesis, it is crucial to both understand the evolution of coding opportunities in IDNC and identify the best packet and receiver selection strategies to maximize these opportunities for subsequent transmissions.

It is worth mentioning here that a strategy may result in a larger number of coding opportunities after a transmission because it served very few packet requests, and thus the number of requests itself would remain also large. Consequently, this apparent large number of coding opportunities will not be actually enough to foster better coding combinations between the large number of remaining requests. Consequently, it is important to not only maximize the number of coding opportunities but to maximize them with respect to the number of remaining packet requests. We refer to this notion as the coding opportunity densification.

According to all the aforementioned problems and concerns of IDNC, this thesis aims to find answers and solutions to the following questions:

1. How can we densify the IDNC coding opportunities after each transmission and over the transmission horizon of a frame of packets?
2. How can we minimize the average completion delay in IDNC for both multicast and broadcast scenarios?
3. How can we adaptively select, between IDNC and FNC, the scheme that is expected to achieve the smaller average completion delay?
4. How can we minimize the average IDNC completion delay in intermittent and lossy feedback scenarios?
Chapter 1. Introduction

The following section includes a summary of the thesis contributions, which provide answers and solutions to these questions.

1.3 Thesis Contributions

This thesis presents four main contributions, summarized in the following four sections.

1.3.1 Contribution 1: Densification of Coding Opportunities in IDNC

In this contribution, we derived expressions for the exact and expected evolution of the IDNC coding opportunities after the transmission of any arbitrary coded packet. From the exact expression, we showed that serving packets requested by the largest number of receivers tends to maximize the density of coding opportunities (or coding density for short), especially in the beginning of the recovery transmission phase. From the expected expression, we showed that targeting the maximum number of worst case receivers (i.e. receivers with the largest number of remaining packet requests and the largest erasure probabilities) with primary packets then secondary packets, tends to maximize the expected coding density along the frame transmission, thus achieving a continuous increase in the coding density. This contribution is detailed in Chapter 4 and its results are to be published in [67,70].

Although our motivation of studying the coding opportunity densification in this thesis is to optimize the IDNC completion delay, the evolution of coding opportunities and density is a very important factor that can greatly affect the optimization of any other IDNC parameter. Thus, this contribution is generally important in IDNC optimization and its results should be observed when any parameter is optimized over the full course of a frame transmission.
1.3.2 Contribution 2: Completion Delay Minimization in IDNC

In this contribution, we first formulated the minimum completion delay problem in IDNC as a stochastic shortest path (SSP) problem and showed that its solution is intractable. Nonetheless, we were able identify the two main components affecting the completion delay IDNC, using the properties of the formulated SSP and the nature of the IDNC graph evolution. We found that targeting the maximum number of worst case receivers with primary then secondary packets can efficiently reduce the average completion delay in IDNC. Based on these guidelines, we designed a maximum weight clique selection algorithm that both efficiently reduces the IDNC completion delay and achieves a near-optimal performance in wireless broadcast. For further complexity reduction, we designed a quadratic-complexity maximum weight vertex search heuristic, which efficiently tracks the optimal clique selection performance. This contribution is detailed in Chapter 5 and part of its results is published in [69]. The whole contribution is to be published in [67].

1.3.3 Contribution 3: Adaptive IDNC-FNC Selection

In this contribution, we designed an adaptive network coding (ANC) scheme that combines the completion delay reduction gains of both IDNC and FNC, by selecting the scheme that is expected to achieve the smaller completion delay. The core of this ANC lies in our derived performance metrics for the IDNC completion delay, which achieve efficient selection when compared to appropriate FNC metrics. To compare between different complexity levels and information requirements, we presented three approaches to derive the IDNC metrics. In each of these approaches, we first modeled the IDNC graph representation by three random graphs with different parameters, using different levels of knowledge about the original graph and erasure probabilities. We then derived an IDNC completion delay metric for each approach using a well-known expression of the random graph chromatic number. All three approaches almost achieve the completion
delay performance that could be obtained by the optimal selection between IDNC and FNC. This contribution is detailed in Chapter 6 and its results are published in [65,66].

1.3.4 Contribution 4: Extensions to Intermittent and Lossy Feedback

In this contribution, we formulated the completion delay minimization problem, for the cases of intermittent and lossy feedback, as extended SSP and partially observable SSP problems, respectively. We then showed that these new formulations have the same structure of the original SSP. Consequently, we tested four different extensions of the designed algorithms in Chapter 5 to operate in intermittent and lossy feedback scenarios, after taking four different update decisions on the attempted and un-acknowledged packet requests. For the intermittent feedback scenarios, the algorithm eliminating all the attempted and un-acknowledged requests during the feedback intermittence achieves the best performance among the four schemes. For the lossy feedback scenarios, the best performance is mostly achieved by the algorithm that stochastically eliminates the attempted and un-acknowledged packet requests, where the probability of eliminating an un-acknowledged request is equal to the reception success probability of its inducing receiver. These extended algorithms were shown to achieve tolerable degradation for relatively low feedback frequencies and high feedback loss rates. This contribution is detailed in Chapters 7 and 8 and its results are to be published in [63,64,71,72].

1.4 List of Publications from Thesis

Journal Papers

• S. Sorour and S. Valaee, “Coding Opportunity Densification Strategies in Instantly Decodable Network Coding,” *submitted to IEEE Transactions on Communications.*

• S. Sorour and S. Valaee, “Completion Delay Minimization for Instantly Decodable Network Coding,” *submitted to IEEE/ACM Transactions on Networking.*


**Conference Papers**


• S. Sorour and S. Valaee, “Completion Delay Minimization for Instantly Decodable Network Coding with Limited Feedback,” *accepted for publication in IEEE International Conference on Communications (ICC’11),* 2011.


Chapter 2

Background Work

In this chapter, we summarize the different works related to the problems of interest in this thesis. We will also briefly illustrate some mathematical tools that we will use in our studies. Before starting this summary, we will first illustrate some important definitions that we will use in this chapter and the rest of the thesis.

2.1 Important Definitions

Definition 2.1 (Source Packets). A source packet is an original packet that has not undergone any network coding operations with other packets.

Definition 2.2 (Coded Packet). A coded packet is a packet that consists of a combination of source packets in the network coding sense. With this definition, a source packet can be viewed as a special case of coded packet with only this source packet in its combination.

Definition 2.3 (Innovative Packet). A coded packet is called innovative to a receiver if its combination is linearly independent from all previously received packet combinations (including source packets). Otherwise, this coded packet is called non-innovative.

Definition 2.4 (Instantly Decodable Packet). A packet is called instantly decodable for a receiver if its combination includes only one source packet that is not previously decoded.
by this receiver. This missing source packet can be thus decoded at the reception instant of the instantly decodable packet by simply eliminating the known source packets. Every instantly decodable packet for a receiver is innovative for this receiver but not vise-versa.

**Definition 2.5** (Non-Instantly Decodable Packet). A packet is non-instantly decodable for a receiver if it includes in its combination two or more source packets that are not previously decoded by this receiver. Every non-instantly decodable packet for a receiver is innovative for this receiver but not vise-versa.

**Definition 2.6** (Informative Packet). A packet is called informative to a receiver if its combination both includes at least one source packet that is wanted by this receiver and is linearly independent from all previously received combinations (including source packets). Otherwise, it is called non-informative. Clearly, every informative packet for a receiver is innovative for this receiver but not vise versa.

**Definition 2.7.** A rate-optimal network coding scheme is a scheme in which all the transmissions are informative for the entire set of receivers.

**Definition 2.8** (Completion Delay). The completion delay is defined as the number of recovery transmissions until all the receivers receive all of their requested packets. In other words, it is the number of transmissions from the beginning of the recovery transmission phase until the last receiver gets its last requested packet.

**Definition 2.9** (Decoding Delay). In any transmission, a receiver experiences one unit increase of decoding delay if it successfully receives a packet that is either non-instantly decodable or non-informative.

**Definition 2.10** (Offline (Non-Causal) Algorithm). An offline algorithm is an algorithm that takes decisions on packet combinations based on a full knowledge of future realizations of all channel erasures for all receivers.
**Definition 2.11 (Online (Causal) Algorithm).** *An online algorithm is an algorithm that takes decisions on packet combinations solely based on the information received after the past transmissions.*

## 2.2 Full Network Coding

FNC is a class of network coding that combines all source packets, which arrive at any node, in each and every coded packet transmission. This encoding process is usually done algebraically [34,35] over finite fields, whose dimensions should be suitable for both the number of source packets and the number of destinations. Other than some studies considering nonlinear network coding [36,43], most of the works in FNC intensively studied linear network coding (LNC) [41].

In LNC, the coded packets are generated by multiplying all the source packets with coding coefficients and adding the resulting terms. All these operations are performed over finite fields. The coding coefficients can be either deterministic or selected from a large field such that a large number of coded packets is guaranteed to be linearly independent almost surely. The latter case, introduced in [24,26] and expanded in [25], has been more studied in the literature and is generally referred to as the random linear network coding (RLNC). Both LNC and RLNC operate by continuously sending coded packets until the receivers collect enough linearly independent coded packets to be able to decode all the source packets. The decoding process is performed using Gaussian elimination [81] and matrix inversion, which is of cubic complexity in the number of source packets [22]. In [47], it has been shown that RLNC can achieve the unicast and broadcast capacities of a network for a large number of source packets.

In [28,29], FNC was employed for packet recovery in WiMAX unicast. In [51], the same idea was extended to wireless broadcast. The delay and throughput gains of RLNC in single-hop wireless networks, with packet erasures, was studied in [18,19]. In [45,46],
RLNC was studied in the context of time division duplex (TDD) with the target of minimizing its completion delay.

In this thesis, we will mainly employ FNC as a scheme to which we compare our proposed IDNC solutions. In our simulations of FNC, we assume the following:

- Our definition of FNC is only restricted to linear network codes as they are sufficient to achieve the optimal full coding performance in our single-hop setting.
- We assume that all FNC packets are linearly independent, which results in the optimal completion delay performance in wireless broadcast.

### 2.3 Opportunistic Network Coding (ONC)

The ONC scheme is a class of network coding that exploits the diversity of received and lost source and coded packets, at different receivers, in opportunistically combining them so as to maximize the number of receivers for which the coded packet is informative. Consequently, ONC requires feedback information about the received and lost packets in order to operate properly. Although several works have considered using linear coding in ONC, most ONC works employ XOR for combining packets, which makes the encoding and decoding processes much simpler.

One subclass of ONC that has recently attracted much attention is the instantly decodable network coding (IDNC). In the literature, the IDNC sender enforces the generated packet combinations to be either non-innovative or instantly decodable for all the intended receivers. In this thesis, we refer to this type of IDNC as the strict IDNC. IDNC uses XOR for packet encoding and decoding as linear coding would not add any benefit to it but will increase its complexity.

The design of offline and online ONC and IDNC algorithms, which minimize different metrics over erasure channels, has lately been an intensive area of research. [31,32] introduced COPE (Coding Opportunistically), a first architecture for XOR-based ONC
designed for packet forwarding in wireless mesh networks. This architecture was implemented on a test bed of 20 nodes and has shown great throughput improvement. In [58], ER (Efficient Retransmission Scheme) was introduced as a modification to COPE for multiple unicast streams, in which packet combinations are restricted to strict IDNC. ER tries to find the minimum number of lossless transmissions for these unicast sessions using an exhaustive search tree. In [40], the concept of multiuser ARQ is introduced and its throughput performance was tested in Rayleigh channels. This test revealed both significant SNR and diversity gains, which increase with the growth in the number of users.

In [50, 52], the authors derived upper-bound performance expressions for ONC in the broadcast scenarios. In [77], these expressions were extended to the case of combining both network and channel coding, similar to the idea of HARQ. However, these three works did not give any insights nor designed any algorithms on how coded packets should be scheduled for their expressions to be true.

In [14, 20, 33, 73–76], online network coding algorithms for random information arrivals were studied. These works assumed that information symbols could be combined dynamically within a sliding window, which is updated according to the number of seen source packets by the different receivers. A "seen" source packet \( j \) at receiver \( i \) is a packet for which receiver \( i \) has enough information to compute a linear combination of the form \((j + l)\), where \( l \) is itself a linear combination involving only packets that arrived after \( j \) at the sender (Decoding implies seeing, as we can pick \( l = 0 \)) [73, 75]. Note that in this work, packets are treated as vectors over a finite field.

In [20], different methods of employing feedback, in systems using network coding, were examined in order to satisfy QoS requirements as well as for reliability purposes. In [76], the authors proposed an ONC algorithm for the three-receiver case, proved its rate optimality and conjectured that it achieves an asymptotically optimal average delay. In this work, the considered delay notion is defined as the time elapsed between the arrival
of a packet at the sender until its decoding at a given receiver averaged over the packets in the long run in a packet streaming scenario.

In [73, 75], a new coding and queue management algorithm is proposed for communication networks that employ sliding widow based linear network coding with random arrivals and packet erasure broadcast channel. The proposed drop-when-seen algorithm guarantees that the physical queue size at the sender tracks the backlog in degrees of freedom and limits the expected queue size to $O\left(\frac{1}{1-\rho}\right)$, with $\rho$ being the load. The impact of this approach on in-order decoding delay at the receivers is studied and more strategies for adaptive coding based on feedback are presented in [73], with the goal of minimizing the queue size and delay. The delay implications of these algorithms were also studied in [6].

In [14, 33], the decoding delay performance of offline ONC algorithms were studied. Some heuristic online algorithms were also compared in case of independent and identically distributed erasure channels. In [33], the authors considered optimizing the decoding delay. They first showed that there exists a polynomial time offline algorithm that can achieve zero worst case delay for three receivers, if it has one future erasure pattern and employs binary operations. They then studied the case of four receivers for a given erasure pattern and derived some bounds on the average and worst case delays achievable by offline algorithms. They also proved that, for any number of receivers, there exists a rate-optimal offline algorithm that can guarantee zero delay for one receiver. For the online algorithms, they first proved that the worst case delay for the case of 3 receivers is lower bounded by $N/2$, with $N$ being the number of packets in the frame. Finally, they compared different heuristic online algorithms using simulations.

In [14], the above work was extended to show some new results, mainly on offline algorithms. The authors first showed that minimizing average and worst case offline decoding delays, when the source uses uncoded packet scheduling, is NP-hard. They then examined the complexity of the problem when coding is allowed and showed that,
although specific classes of erasure instances become trivial, the general problem remained
NP-hard. However, they showed that coding offered significant gains in delay for some
classes of erasure instances. In both [14,33], the studied online algorithms selected packets
in an un-prioritized fashion for each NC transmission and did not consider the channel
conditions in their selection procedures.

[12] considered several online IDNC heuristics that minimize the decoding delay and
analyzed their performance by means of simulation. The studied algorithms differed both
in the required information about the state of the neighbors’ buffers and in the way this
knowledge is used to decide which packets to combine in each transmission. Simulation
results showed that serving the maximum number of receivers significantly outperforms
the algorithms in [14,33]. However, these heuristics were not designed according to any
theory supporting their design, nor considered the effect of these decisions on future
steps. In [37], a novel MAC-layer retransmission scheme employs network coding aware
fair opportunistic scheduling for multiple unicast sessions, which is theoretically proven
to be fair. Moreover, it devises a new coding metric, which accommodates the effects of
the frames size and the channel condition. Finally, it presents a first attempt to estimate
the reception status of secondary packets without collecting their feedback.

In [59,60], the minimum decoding delay problem for strict IDNC was formulated as
a set packing problem [8] and optimal and heuristic algorithms were designed to solve
it. This work provided more structured improvements on the heuristic algorithms given
in [14,33], by giving priority to the packets that are needed by most receivers, instead
of using a random packet selection strategy. The proposed algorithms were shown to
outperform the equivalent random selection algorithm in [14,33]. They also extended the
problem to the case of Gilbert-Elliott erasure channel model, in which the quality of the
channel is modeled by Markovian transitions between Good (full reception) and Bad (no
reception) states.

Our work can be looked upon as a multi-level extension to [59,60] as follows. We
extend the prioritized receiver and packet selection for network coding to optimize a more challenging parameter, namely the completion delay. Unlike decoding delay optimization, the completion delay is not a parameter that can be locally optimized for each transmission. We also extend our study of the problem to the general multicast scenario, which brings an added challenge in optimizing both decoding and completion delay. Moreover, we consider the optimization of completion delay in more harsh feedback scenarios, such as intermittent and lossy feedback.

2.4 Index Coding

The index coding problem was studied in several works [5, 11, 15, 16] motivated by applications in wireless networking and distributed computing. It includes a sender and a set of receivers and error-free channels between the sender and these receivers. Each receiver already has a subset of the packets, held at the sender, and requests to receive another subset of them. In each transmission, the sender can send either a source or coded packet. The objective of the index coding problem is to define the packet coding schedule that delivers the requested packets by all receivers with the minimum number of transmissions (i.e. minimum completion delay). The coding gain achieved by index coding is defined as the ratio of the number of transmissions needed to satisfy all receivers without encoding, to the minimum number of transmissions required when encoding is employed.

In [5, 16], it has been shown that finding the optimal solution of the index coding problem is NP-hard. Consequently, different heuristics to solve the index coding problem were proposed in [11]. These heuristics define a graph in which each vertex $v_{ij}$ represents a request of packet $j$ by receiver $i$. The coding opportunity between any two vertices is represented by an edge in the graph. Consequently, the minimum number of transmissions in this heuristic can be obtained by performing a clique partitioning on
In [58], an exhaustive search algorithm on a coding tree was developed to minimize the recovery transmissions of strict IDNC over lossless channels. This developed algorithm is found to be computationally complex as it exhaustively searches over all packet combinations in every transmission.

The above works optimize the same parameter we are considering in this thesis. However, this optimization is done over lossless channels. Despite the complexity of these problems, they remain very different in nature and complexity than the problem of optimizing the same parameter over erasure channels. In the lossless channel scenarios, all the transmissions that minimize the completion delay are selected using clique partitioning on one graph at the beginning of the recovery phase. In other words, the sender performs one decision on one graph to generate all the required cliques in the beginning of the recovery phase. This decision is not about receiver and packet scheduling but rather about a clique partitioning to resolve the decodability conflicts using the smallest number of cliques.

In the lossy channel case, the graph is much more dynamic and changes after each transmission, according to the selected clique for this transmission and its reception at its intended receivers. In this case, the sender needs much more decisions, each on a different graph, which makes the packet and receiver selection and scheduling a major concern in order to minimize the completion delay. This makes this problem a totally different problem than the one assuming lossless channels.

### 2.5 Markov Decision Processes (MDP)

A Markov decision process (MDP) is a discrete time stochastic control process that provides a mathematical framework for modeling decision-making in dynamic systems where outcomes of the actions decided by the decision-maker are random [57]. At each
time step, the process is in some state $s$, and the decision maker may choose any action $a$ in state $s$ that results in an immediate cost $c(s, a)$. The process then responds to this action by randomly moving into a new state $s'$. The probability $P_a(s, s')$ that the process moves to state $s'$ as its new state depends only on the current state $s$ and action $a$, which makes the process Markovian.

According to the above description, an MDP is generally defined by four-tuples:

1. State space $\mathcal{S}$: It represents all the possible situations the process can go through.
2. Action space $\mathcal{A}$: It represents the set of all possible actions in the different states of the process. In some processes, it is more comprehensive to define an action space $\mathcal{A}(s) \in \mathcal{A}$ for each state $s \in \mathcal{S}$.
3. State-action transition probability space $\mathcal{P} = \{P_a(s, s')\}$: Each of these probabilities represents the transition probability from state $s$ to state $s'$ when action $a$ is taken.
4. State-action cost functions $c(s, a)$: It represents the immediate cost endured by the process when taking action $a$ in state $s$.

A policy $\pi = [\pi(s)]$ is a mapping from $\mathcal{S} \rightarrow \mathcal{A}$ that specifies a given action to each of the process states. When a policy $\pi$ is attributed to an MDP, it transforms it into a Markov chain with state transition probability matrix $P_\pi$. The main problem in MDP is to find an optimal policy $\pi^*$ that minimizes:

- The overall expected cost over a finite horizon, when the process will stop after a finite number of steps.
- The expected cost per unit time (or expected discounted cost) over an infinite horizon, when the process continues indefinitely.

In general, the action space, transition probabilities and costs may be either stationary or time-varying, which leads to either stationary or non-stationary optimal policies. In this
thesis, we will focus only on stationary MDPs and optimal policies as they are enough to fit our problems. We will also consider un-discounted MDPs.

The family of algorithms to calculate the optimal policy for an MDP defines two vector variables with size $|S| \times 1$:

- The value function vector $\mathbf{V}_\pi = [V_\pi(s)]$, where $V_\pi(s)$ is defined as the expected cumulative cost (for finite horizon MDPs), or expected cost per unit time (for infinite horizon MDPs) the process endures if it started at state $s$ and followed policy $\pi$.

- The policy vector $\pi = [\pi(s)]$, where $\pi(s)$ is the action taken at state $s$ as enforced by policy $\pi$.

During the run of the solving algorithm, the values in vector $\mathbf{V}_\pi$ and $\pi$ are updated iteratively until the algorithm converges to the optimal value and policy vectors $V_{\pi^*}$ and $\pi^*$. The algorithms follow two kinds of update steps, which are repeated in some order for each state $s \in S$ until no further changes take place:

\begin{align}
V_\pi(s) &= c(s, \pi(s)) + \sum_{s' \in S(s,a)} P_{\pi(s)}(s, s') V_\pi(s') \quad \text{(2.1)} \\
\pi(s) &= \arg \min_{a \in A} \left\{ c(s, a) + \sum_{s' \in S(s,a)} P_a(s, s') V_\pi(s') \right\} \quad \text{(2.2)}
\end{align}

Note that the set of equations in (2.1) are the Bellman equations for stochastic dynamic systems. The most two well-known algorithms to find the optimal policy of an MDP are the policy iteration algorithm (PIA) and the value iteration algorithm (VIA) (also known as the backward induction algorithm (BIA)). In PIA, the algorithm first assumes an initial policy then solves the system of linear equations in (2.1). This step is called the value determination step, with complexity $O(|S|^3)$ [44]. In the next step, the algorithm updates the policies minimizing the new value function expressions using the set of equations...
in (2.2). This step is referred to as the *policy improvement step*, with a complexity $O(|S|^2|A|)$ [44]. These two steps are repeated until no further change in the policy is possible. The obtained value and policy vectors are thus optimal. According to the above description, the overall complexity of one iteration in the PIA is $O(|S|^3 + |S|^2|A|)$. In general, the number of iteration for PIA to converge in polynomial time in $|S|$ and $|A|$ [44].

In VIA, value functions are first initialized then are updated iteratively $\forall s \in S$ as follows:

$$V^{(k)}(s) = \min_{a \in A} \left\{ c(s, a) + \sum_{s' \in S(s, a)} P_a(s, s') V^{(k-1)}(s') \right\}$$ (2.3)

The iterations of the algorithm continues until the variation in value functions drops below a small tolerance value. Consequently, the complexity of each iteration in the VIA is $O(|S|^2|A|)$ [44]. It has been shown that running the VIA for a number of iterations, which is polynomial in $|S|$ and $|A|$, guarantees the convergence to the optimal policy [44].

## 2.6 Stochastic Shortest Path (SSP) Problem

The stochastic shortest path (SSP) problem is a special case of the infinite horizon MDP [57], where the process has one zero-cost absorbing state $s_0$ at which it terminates (i.e. $P_a(s_0, s_0) = 1$ and $c(s_0, a) = 0 \forall a \in A$). A solution for an SSP is the optimal policy that reaches to the absorbing state with the minimum expected cost from any other state. The existence of such optimal policy is guaranteed under the following conditions:

- There exists a policy that ensures that the state is reached with probability 1 from any starting state. Such policy is called a *proper* policy.

- All costs are positive (except for that of the absorbing state which is zero).

If these conditions hold, than the PIA and VIA can find an optimal policy for SSP.
Quite often, the decision maker is only interested in how to get to the absorbing state from a fixed starting state \( s \) instead of knowing the general solution (this applies to our case). For such approaches, different algorithms have been proposed such as state-space search algorithms (such as A*, AO* [10] and LAO* [23]) and Real-Time Dynamic Programming (RTDP) [7,42]. However, if this state is the furthest in terms of transitions to the absorbing state, all these algorithms will still scale with the entire state space size because they will need to compute the policies \( \pi(s) \forall s \in S \) as they all can be reached during the transition from this furthest starting state until the absorbing state is reached.

### 2.7 Partially Observable MDP (POMDP)

A Partially Observable Markov Decision Process (POMDP) is a generalization of MDP, in which the decision maker cannot directly observe the reached state after each action. Instead, it must maintain a probability distribution over the set of possible states, based on a set of observations and observation probabilities. This probability distribution is called belief.

A POMDP can be modeled by adding two tuples to the four tuples of MDP:

1. **Observation space** \( \mathcal{O} \): It represents the set of all observations that can occur in the system. We may also define a specific observation space \( \mathcal{O}(a) \in \mathcal{O} \) for each action \( a \).

2. **Conditional observation probability space** \( \mathcal{Q} = \{Q_a(s,o)\} \): Each of these probabilities represents the probability of observing observation \( o \) after taking action \( a \) and moving to state \( s \).

Since we cannot perceive the state transitions in POMDP, we define a belief state \( b = [b(s)] \forall s \in S \) as a probability distribution over all the states of the POMDP where \( b(s) \) is the probability that the process is in state \( s \). All the possible belief states that
can be encountered in the process constitute the belief space \( \mathcal{B} \). If the process is thought to be at a belief state \( \mathbf{b} \), the entries of the new belief state \( \mathbf{b}' \), after taking action \( \mathbf{a} \) and observing observation \( \mathbf{o} \), can be computed as:

\[
\mathbf{b}'(s') = U(b(s), a, o) = \frac{Q_a(s', o) \sum_{s \in \mathcal{S}} P_a(s, s') b(s)}{\Omega_a(b, o)},
\]

(2.4)

where \( \Omega_a(b, o) \) is defined as:

\[
\Omega_a(b, o) = \sum_{s' \in \mathcal{S}} \left( Q_a(s', o) \sum_{s \in \mathcal{S}} P_a(s, s') b(s) \right)
\]

(2.5)

In the above expression, the function \( U(b(s), a, o) \) is called the belief update function for the state \( s \). We can also write this update function in vector format for the entire belief state as \( \mathbf{b}' = U(\mathbf{b}, a, o) \). Since the state is Markovian, maintaining a belief over the states solely requires knowledge of the previous belief state, the action taken, and the current observation.

The policy of a POMDP maps the current belief state into an action. As the belief state holds all relevant information about the past, the optimal policy of the POMDP is the solution of (continuous-space) belief MDP defined as follows:

1. The state space of the belief MDP is the belief space itself. In other words, each state in the belief MDP corresponds to a belief the process can have.

2. The action space is the same as the action space of the original POMDP. Each belief state \( \mathbf{b} \) can have a set of actions \( \mathcal{A}(\mathbf{b}) \).

3. The belief-action transition probability space \( \mathcal{P}^b = \{ P_a(\mathbf{b}, \mathbf{b}') \} \): Each of these probabilities represent the transition probability from belief state \( \mathbf{b} \) to belief state


\[ \text{Chapter 2. Background Work} \]

30
b’ when action a is taken. This probability can be expressed as:

\[ P_a(b, b’) = \sum_{o \in O | U(b, a, o) = b’} \Omega_a(b, o). \tag{2.6} \]

4. The belief-action costs \( c(b, a) \) represent the expected cost endured by the process if it was in belief \( b \) and made action \( a \). These costs can be computed as:

\[ c(b, a) = \sum_{s \in S} b(s)c(s, a). \tag{2.7} \]

The partially observable SSP (PO-SSP) [54, 55] is a special case of POMDP having one or several states are absorbing states. The relation between PO-SSP and POMDP is similar to that of SSP to MDP.

### 2.8 Online NC Algorithms using MDP and POMDP

To the best of our knowledge, the first approach to model online network coding transmissions as an MDP is done by Nguyen et al. in [49]. This work aimed to design an optimized scheduling algorithm, to maximize the quality of multimedia applications over a finite horizon of \( K \) slots. Consequently, it modeled the coded packet scheduling using a finite horizon MDP. The immediate rewards (which can be seen as negative costs) of the MDP was modeled as the sum of the reduction in media distortion for one or more receivers upon receiving a particular source or coded packet. The media distortion represents a reduction in quality of the received media, such as lower quality image in layered video. However, the authors did acknowledge the extremely large state and action spaces of the problem model, which grow exponentially with the number of receivers and packets per frame. They actually presented a simulative solution for only 2 receivers and 2 packets, using the BIA. This makes the state and action spaces sufficiently small to be
tractable (16 states and 3 actions). Note that the complexity of the BIA for a K-horizon MDP is $O(K|S|^2|A|)$.

In [48], the same authors formulated the same problem for lossless and lossy feedback channels using, MDP and POMDP, respectively. They also presented a simulation-based dynamic programming (SDP) algorithm, where actions are sampled for $n$ times using simulations at each step instead of exactly solving the BIA. For each state, the action that achieves a higher reward through the sampling process is chosen as the policy for this state. The complexity of this algorithm is $O(nK|S||A|)$, which is much less than BIA, but still is intractable for large state and action spaces.

Despite the modeling of the optimization parameter (multimedia quality) in these two works using MDP, they did not study the use of the modeled MDP structure to design more efficient real-time algorithms for large numbers of receivers and packets. Moreover, only one parameter has been studied in this sense and many other parameters can be studied similarly to design efficient network coding algorithms.
Chapter 3

Instantly Decodable Network Coding

3.1 Different Types of IDNC

In the literature, the term instantly decodable network coding always refers to a sub-class of opportunistic network coding in which the sender generates XOR-based packet combinations that must be either non-innovative or instantly decodable for all their intended receivers. In this way, no packets will be useful by any chance after their reception instant. The reason this class has been attractive to researchers is its numerous attractive properties and implementation benefits that are most desirable in several applications and attractive from the deployment viewpoint, such as:

1. It allows fast and progressive decoding of packets at the receivers, which reduces their reception delays. This property is of great importance to applications requiring progressively refined input, without long delays.

2. It employs simple binary XOR for packet encoding at the sender, thus eliminating complicated coding operations over large Galois fields and eliminating the coefficient reporting overhead.
3. It enables a simple $O(N^2)$ XOR decoding at the receivers, thus eliminating the need for computationally expensive matrix inversions at the receivers.

4. It requires no buffers to store non-instantly decodable packets for future decoding opportunities.

The $O(N^2)$ decoding complexity of IDNC can be explained as follows. Since an instantly decodable packet for a receiver includes only one new source packet for this receiver in its combination, the XOR decoding is carried on by re-XORing all known source packets to the instantly received coded packet, in order to extract the new source packet from the combination. This requires $O(N)$ XOR operations. Since the number of recovery transmissions is $O(N)$, the overall complexity of decoding the whole frame is $O(N^2)$, which is much smaller than the cubic decoding complexity for linear network coding. This property along with the needlessness for decoding buffers allows the implementation of simple and cost-effective receivers, which is a favorable property for the proliferation of the technology.

According to the description of this IDNC type, we can refer to it as the strict IDNC (S-IDNC), since it imposes strict conditions on the generated packet combinations. We can also call this type as the sender driven IDNC, since the instant decodability constraint is enforced by the sender.

In this thesis, we extend the definition of instantly decodable network coding to a more general definition as being the sub-class of opportunistic network coding, in which coded packets are allowed to be decoded only at their receipt instant, and cannot be stored for future decoding. According to this definition, we can envision a more general type of IDNC, which relaxes the instant decodability constraint at the sender, but forces the receivers to discard all packet combinations including more than one of their missing source packets. In this way, the encoding and decoding processes can still be done in binary XOR and the receivers will not need decoding buffers. Although this scheme loosens the instant decodability constraint at the sender, all the decoding processes at
the receivers are still from instantly received packets. Consequently, we will refer to this scheme as \textit{generalized IDNC (G-IDNC)}, since it considers a more general understanding of the instant decodability property. It can also be referred to as the \textit{receiver driven IDNC}, since the instant decodability property is imposed by the receivers. It is important to note here that this receiver driven IDNC approach was previously employed in one form or another in other works with different names, such as [12,37], and was studied to optimize other parameters as explained in Section 2.3. However, the study of this approach as part of IDNC is novel to us as none of the other works refers to it as IDNC nor studied its completion delay.

3.2 Packet and Receiver Classifications in IDNC

In order to determine IDNC packet combinations, we first define the following packet and receiver sets that are generated at the end of the initial transmission phase, and are maintained and updated along the recovery transmission phase.

- The \textit{Has} set (denoted by $\mathcal{H}_i$) of receiver $i$ is defined as the set of primary and secondary packets correctly received by receiver $i$. This set includes both desired and undesired packets by this receiver.

- The \textit{Lacks} set (denoted by $\mathcal{L}_i$) of receiver $i$ is defined as the set of primary and secondary packets that were not correctly received by $i$. In other words, $\mathcal{L}_i = \mathcal{N} \setminus \mathcal{H}_i$.

- The \textit{Wants} set (denoted by $\mathcal{W}_i$) of receiver $i$ is defined as the set of primary packets of receiver $i$, which are not yet received by it. Note that $\mathcal{W}_i \subseteq \mathcal{L}_i$.

- The \textit{Received} set (denoted by $\mathcal{R}_j$) of packet $j$ is defined as the set of receivers that received packet $j$.

- The \textit{Un-received} set (denoted by $\mathcal{U}_j$) of packet $j$ is defined as the set of receivers that have not received packet $j$. 
• The Demands set (denoted by $D_j$) of packet $j$ is defined as the set of receivers that still demand packet $j$. In other words, it is the set of receivers that have packet $j$ in their Wants sets.

The sender stores this information in a state feedback matrix (SFM) $F = [f_{ij}]$, $\forall i \in \mathcal{M}, j \in \mathcal{N}$ such that:

$$f_{ij} = \begin{cases} 0 & j \in \mathcal{H}_i \\ 1 & j \in \mathcal{W}_i \\ -1 & \text{otherwise} \end{cases} \quad (3.1)$$

Define $\varrho = [\varrho_1, \ldots, \varrho_M]$, $\varphi = [\varphi_1, \ldots, \varphi_M]$, $\psi = [\psi_1, \ldots, \psi_M]$ and $\zeta = [\zeta_1, \ldots, \zeta_N]$ as the Has, Lacks, Wants and Demands vectors, such that $\varrho_i$, $\varphi_i$, $\psi_i$ and $\zeta_j$ are the cardinalities of $\mathcal{H}_i$, $\mathcal{L}_i$, $\mathcal{W}_i$ and $\mathcal{D}_j$, respectively.

### 3.3 IDNC Graph Representation

In order to optimize different parameters in IDNC, we should first define a representation of all feasible packet combinations, which are instantly decodable by any subset or all the receivers, in the more general multicast scenarios. In this section, we describe and compare the graph representations of packet combinations for G-IDNC and S-IDNC. We will first start with G-IDNC as it has more relaxed packet combination rules than S-IDNC.

#### 3.3.1 G-IDNC Graph

An initial idea for the representation of packet combinations in a setting similar to G-IDNC was introduced in [11,16] in the form of a graph, as part of designing a heuristic algorithm to solve the index coding problem. In this chapter, we build on this graph representation and extend it to the more general multicast scenarios. We will refer to
this graph as the G-IDNC graph.

The graph employed in [11,16], which we will denote by $G_\rho(V_\rho,E_\rho)$, is constructed by first inducing a vertex $v_{ij}$ in $V_\rho$ for each packet $j \in \mathcal{W}_i$, $\forall i \in \mathcal{M}$. Obviously, the vertex $v_{ij}$ represents a request from receiver $i$ of packet $j$. Two vertices $v_{ij}$ and $v_{kl}$ in $V_\rho$ are adjacent by an edge in $E_\rho$ if one of the following conditions is true:

- **C1**: $j = l \Rightarrow$ The two vertices are induced by the loss of the same packet $j$ at two different receivers $i$ and $k$.

- **C2**: $j \in \mathcal{H}_k$ and $l \in \mathcal{H}_i \Rightarrow$ The requested packet of each vertex is in the Has set of the receiver that induced the other vertex.

Consequently, each edge between two vertices in the graph represents a coding opportunity that is instantly decodable for the receivers inducing these vertices. Given this graph formulation, we can easily define the set of all feasible packet combinations in IDNC as the set of packet combinations defined by all cliques in $G_\rho$. Consequently, the sender can generate an IDNC packet for a given transmission, by XORing all the packets identified by the vertices of a selected clique $\kappa_\rho$ in $G_\rho$.

The above formulation of $G_\rho$ is suitable when optimizing packet combinations in broadcast settings. However, the optimization of packet combinations in multicast scenarios necessitates the delivery of all possible secondary packets to the receivers that are not considered for primary packet reception. Although these packets are not required to be received at these receivers, their reception throughout the steps of the recovery transmission phase will enlarge their Has sets. This will increase the coding opportunities for these receivers in the future steps, as mandated by Condition C2. However, the inclusion of these packets in each step should not affect the instant decodability of the primary packets at the other receivers.

To achieve both goals, we propose a new two-layered graph design $G(V,E)$. The primary layer consists of the graph $G_\rho$, described above. The secondary layer $G_\sigma(V_\sigma,E_\sigma)$
is constructed by generating a vertex $v_{ij} \in V_\sigma$ for each packet $j \in L_i \setminus W_i$, $\forall \ i \in M$, and connecting any two vertices $v_{ij}$ and $v_{kl}$ with an edge if either Condition C1 or C2 holds. Finally, we connect two vertices $v_{ij}$ and $v_{kl}$ from both layers with an edge if either Condition C1 or C2 holds. In the rest of the paper, we will call $G_\rho$, $G_\sigma$ and $G$ as the G-IDNC primary, secondary, and complete graphs (or G-IDNC graph for short if it is clear from the context), respectively. We also define $T_\rho(\kappa)$, $T_\sigma(\kappa)$ and $T(\kappa)$ as the sets including all the primary, secondary and overall targeted receivers by a given clique $\kappa$. We refer to any edge connecting any two primary vertices (i.e. any edge within the primary graph) as a primary edge. We finally define the sets $K$ as the sets of all feasible packet combinations (i.e. all cliques) in the G-IDNC complete graph.

Figure 3.1 depicts an example of a feedback matrix and its corresponding G-IDNC graph. The shaded and white boxes (vertices) in the matrix (graph) represent the requested and undesired packets, respectively.
Inside the set of all feasible packet combinations $\mathcal{K}$, we can define the subset $\mathcal{K}_m$ of all packet combinations, which cannot be encoded with any additional source packet, without violating the instant decodability property for the receivers they are targeting. This set consists of all the maximal cliques in $\mathcal{G}$, where a maximal clique is a clique that is not a subset of any larger cliques. This set will be more favorable in our study as its combinations do not ignore any possible coding opportunities in any transmission. However, the more general set will be also useful in some scenarios as will be seen later.

We can easily infer that each maximal clique $\kappa$ in $\mathcal{G}$ can include at most one primary or secondary vertex induced by any given receiver to maintain instant decodability. Consequently, the selection of a maximal clique of size $n$ at any transmission can be regarded as the selection of a set of $n$ receivers to be targeted by this transmission. This property is usually not valid for packets as several vertices in a clique can represent the same missing packet from different receivers.

### 3.3.2 S-IDNC Graph

The strict instant decodability constraint of S-IDNC in case of wireless broadcast was defined in [59, 60] as follows:

$$ Fx \leq 1, \quad (3.2) $$

where $x$ is the packet selection vector, in which $x_j = 1$ if packet $j$ is selected in the coding combination and zero otherwise. This constraint can be added to any optimization of $x$ in order to guarantee strict instant decodability in the broadcast scenario. This same constraint can be used in the multicast scenarios if the entries of $F$ representing missing but undesired packets (i.e. packets in $\mathcal{L}_i \setminus \mathcal{W}_i \forall i$) are changed from $-1$ to any value greater than 1.

In other words, S-IDNC was represented for the unicast and broadcast scenarios as a graph with one vertex $v_j$ for each packet $j$ if $\mathcal{D}_j \neq \emptyset$ [58]. Two vertices $v_j$ and $v_l$
are adjacent if and only if $D_j \subseteq R_l$ and $D_l \subseteq R_j$. It is clear from this condition that the addition of a packet to a coding combination cannot be done unless all the receivers wanting this packet have all the other packets of this combination, which conforms with the strict instant decodability condition. Indeed, no two packets can be encoded together if only one receiver wants both. This same approach can be extended to the multicast scenarios by replacing $D_j$ and $D_l$ in the graph adjacency condition by $U_j$ and $U_l$.

In order to compare the S-IDNC and G-IDNC approaches, the following lemma introduces another S-IDNC graph representation using the same setting employed for the G-IDNC graph in Section 3.3.1.

**Lemma 3.1.** For any given feedback matrix, all coding combinations of S-IDNC can be formulated as a graph that is a subgraph of the G-IDNC graph.

**Proof.** For any given feedback matrix, we can generate a graph $G_s$ for the S-IDNC, which will have the same vertices of the corresponding G-IDNC graph $G_g$ (constructed from the same feedback matrix). Since packet combinations are represented by the edges in the graph, we can represent the S-IDNC constraint in (3.2) in the vertex adjacency conditions.

For Condition C1 ($j = l$) in Section 3.3.1, the edges do not represent coding combinations since both end vertices of such edges request the same source packet. Consequently, a transmission based on any such edge only will never be non-instantly decodable for any receiver because the transmitted packet is already a source packet. Thus, the edges representing C1 in the G-IDNC graph are all found in the S-IDNC graph.

On the other hand, an edge generated by Condition C2 between $v_{ij}$ and $v_{kl}$ involves a packet combination $j \oplus l$. To ensure that such a combination will never generate a packet that is non-instantly decodable for any receiver, we must add another constraint on this condition to ensure that $j$ and $l$ are not both in the Lacks set of any receiver. In
other words, C2 can be modified for $\mathcal{G}_s$ to be:

$$C2 : j \in \mathcal{H}_k \text{ AND } l \in \mathcal{H}_i \text{ AND } \{j, l\} \notin \mathcal{L}_m \forall m \in \mathcal{M}.$$  \hfill (3.3)

Consequently, the S-IDNC graph $\mathcal{G}_s$ consists of the same vertices of $\mathcal{G}_g$, its same edges generated by C1 and a subset of its edges generated by C2. We can thus conclude that $\mathcal{G}_s$ is a subgraph of $\mathcal{G}_g$ for any given feedback matrix. \hfill \Box

Figure 3.2 depicts the S-IDNC graph for the same example in Figure 3.1. It can be clearly seen that this S-IDNC graph is a subgraph of the corresponding G-IDNC graph in Figure 3.1. We can also see that all edges in the graph represent non-coding opportunities as each edge represents the transmission of one packet. This result shows how S-IDNC can greatly limit the coding capabilities compared to G-IDNC.

This limitation of coding capabilities in S-IDNC can greatly increase its completion and decoding delays compared to G-IDNC. Consider the following example of feedback
matrix:

\[
F = \begin{pmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
-1 & 1 & 1 & 0 \\
0 & 1 & 0 & -1 \\
1 & 0 & -1 & 0
\end{pmatrix}.
\]

(3.4)

It can be easily seen that packets 1, 2 and 3\(\oplus\)4 are the only ones satisfying the S-IDNC constraint. Since each of these packets targets three receivers, each of the other two receivers will experience a decoding delay increase, if it successfully receives the packet. On the other hand, the coded packet 1\(\oplus\)2, violating the S-IDNC constraint, targets four receivers, and thus only one receiver can suffer from a decoding delay increase. Consequently, this packet, which belongs to the G-IDNC type, is expected to achieve a smaller sum decoding delay than those constrained by S-IDNC. In [68], we showed that G-IDNC achieves a much better average decoding delay performance than S-IDNC.

We can also notice from this example that we need to transmit three packets (1, 2 and 3\(\oplus\)4) in S-IDNC to complete the requests of all receivers, which causes the completion delay to be 3 assuming lossless channels. On the other hand, G-IDNC can complete the requests of all receivers in only two lossless transmissions of packets 1\(\oplus\)2 and 3\(\oplus\)4. From this example, we can clearly infer that G-IDNC is a much more promising technique, if we want to significantly reduce the IDNC completion delay to a level that competes with FNC.

### 3.3.3 Complexity of Graph Construction and Update

In this section, we aim to determine the complexity of building and updating the IDNC graph representation, in order to evaluate the overall complexity of our proposed solutions in later chapters. Since we are more interested in G-IDNC, as explained in the previous section, we will derive the complexity of the G-IDNC graph construction and update.
However, a similar approach can be used to determine the complexity of the S-IDNC graph construction and update.

From the description of the G-IDNC graph in Section 3.3.1, we first need to generate $O(MN)$ vertices, representing the different packet loss cases from different receivers. To build the adjacency matrix of the graph, we need to check the adjacency conditions $C_1$ and $C_2$ for each pair of vertices to determine whether they should be connected with an edge. This means that we need at total of $O(M^2N^2)$ operations to build the adjacency matrix.

However, we can reduce this complexity but exploiting the following properties of the adjacency conditions between the vertices of each pair of receivers.

- If $j \in \mathcal{L}_k$, $v_{ij}$ cannot be adjacent to any vertex of receiver $k$, due to violation of $C_2$, except for vertex $v_{kj}$ that satisfies $C_1$.

- If $j \in \mathcal{H}_k$, $v_{ij}$ can be adjacent to any vertex of receiver $k$ (induced from $\mathcal{L}_k$) according to $C_2$, except for all vertices $v_{kl}$ for which $l \notin \mathcal{H}_i \Rightarrow l \in \mathcal{L}_i$.

Consequently, the adjacency matrix can be built as follows. For each pair of receivers, we first find their common lacked packets (i.e packets belonging to the Lacks sets of both receivers), which requires an $O(N)$ search. For these common lacked packets, we link each two vertices $v_{ij}$ and $v_{kj}$ with an edge. The vertices representing non-common lacked packets from both receivers are all connected to each other, resulting in a bipartite full subgraph. Since we have to execute the above action for every pair of receivers, the overall complexity reduces to $O(M^2N)$.

Another important remark is that the construction of the graph can be done on steps along the initial transmission phase. Once a packet is transmitted and its feedback is received, all vertices representing the loss of this packet are directly linked with each other. This requires $O(M)$ operations, since we can get a most $M$ new vertices after each transmission. Afterwards, the new vertex of any receiver is linked to all the vertices of
the receivers that received this packet, except for the vertices representing the previous loss of common packets with this receiver. This requires $O(M)$ checks for common lacked packets with different receivers. Since this step is done for each of the $O(M)$ new vertices, the entire adjacency update can be executed in $O(M^2)$ operations after each initial transmission, and thus the overall graph construction complexity at the end of the initial transmission phase becomes $O(M^2N)$.

For graph update during the recovery transmission phase, the sender removes all vertices whose requested packets have been received by their inducing receivers, which results in $O(M)$ complexity, since this is the maximum number of vertices in a clique as explained in Section 3.3.1. Afterwards, the vertices adjacent to any disappearing vertex $v_{ij}$, according to C1, will be linked to all other vertices of $i$ according to C2. The vertices adjacent to $v_{ij}$ according to C2 will not experience any new edges to other vertices of $i$. This again will incur an overall complexity of $O(M^2N)$.

Finally, we mention that the complexity of building or updating the graph is experienced by the sender, and thus does not represent a severe problem due to the high processing power and energy abilities of senders. On the other hand, this complexity comes as a cost of simplifying both the coding complexity at the sender and most importantly the receivers’ complexity and buffer requirements, when compared to FNC. The reduction in receivers’ requirements is always a much more important target compared to reducing that of the sender’s.

### 3.4 Conclusion

In this chapter, we extended the definition of IDNC to fit another type, which loosens the strict instant decodability constraint and forces the receivers to discard all non-instantly decodable packets. This type that we referred to as G-IDNC provides much better coding opportunities while preserving the attractive properties of S-IDNC. Starting
from a very simple and small feedback matrix, we were able to show that G-IDNC will achieve a smaller completion and decoding delays than S-IDNC. This difference in delay performance is expected to be much larger if the number of receivers and packets is increased.

Based on these results, and since we are aiming to minimize the completion delay in IDNC, we will focus in the rest of the thesis only on G-IDNC. For ease of notation and description, we will use the term IDNC to describe G-IDNC and its graphs in the rest of thesis.
Chapter 4

IDNC Graph Evolution

4.1 Motivation

As stated in Chapter 1, our target in this thesis is to optimize the completion delay of IDNC. Due to the nature of completion delay as a parameter, it cannot be optimally controlled through optimizing the local network coding decision in each transmission, regardless of its effect on future coding opportunities and decisions. It rather requires the optimization of the network coding decisions along the entire horizon of the recovery transmission phase. Consequently, it is important to first study the effect of network coding decisions on the availability of efficient coding opportunities in subsequent transmissions. Indeed, a network coding selection at one given transmission may be serving a large number of receivers for this transmission, which gives hope of faster completion. However, this decision may also result in less efficient coding opportunities for next transmissions, which results in slower steps towards completion. On the other hand, a sub-optimal decision at the first transmissions may result in more numerous and efficient coding opportunities in future steps, which may help in faster completion.

After introducing our extended IDNC graph design in Chapter 3, we identified the coding opportunities between packet requests as the edges of the graph. In this chapter,
we will investigate the properties of the IDNC graph and study the effect of any arbitrary IDNC transmission on the coding opportunities of the resulting graph after this transmission. In this study, we aim to identify the packet and receiver selection strategies that can maximize the coding opportunities in the IDNC graph after each transmission and along the entire horizon of the recovery transmission phase. We will then employ these strategies in our optimization of the IDNC completion delay in the next chapter. Since the completion event occurs once the IDNC primary graph is depleted, regardless of the secondary graph, we will focus on studying the evolution of coding opportunities inside the primary graph only. However, similar results can be derived for the coding opportunities in the IDNC complete graph, as will be shown later.

4.2 Coding Opportunities and Coding Density

Since we study the graph evolution and the maximization of coding opportunities in order to master the tools of minimizing completion delay in IDNC, it is important to notice that determining the strategies that maximize the absolute number of coding opportunities inside the IDNC primary graph may be misleading in several situations. Indeed, a strategy A may result in a larger number of coding opportunities than another strategy B after a transmission because A served very few vertices compared to B, and thus the number of remaining vertices for A will also be larger compared B. Consequently, this apparent large number of coding opportunities in A’s resulting graph will not be actually enough to foster a better adjacency between its large number of remaining vertices. Consequently, it is important to not only maximize the absolute number of coding opportunities but to densify these coding opportunities with respect to the number of the remaining vertices. We refer to this notion as the coding opportunity densification problem. In this problem, we aim for a counter-direction graph densification approach, compared to the one in [56], in which the number of remaining vertices is reduced much
faster than the number of remaining edges.

In order to properly characterize the densification of the coding opportunities, we define the coding opportunity density (or coding density for short) of an IDNC primary graph as the density of its primary graph $G_\rho$. In graph theory, the graph density is the ratio of the total number of edges in the graph to the maximum number of edges that can be found in this graph, given the same number of and type of the vertices. We can thus express this coding density as:

$$\rho_c(G) = \frac{|E_\rho|}{\frac{1}{2}|V_\rho|(|V_\rho| - 1)}$$

From the above expression, the maximization of coding density guarantees a larger number of coding opportunities with respect to the number of remaining packet requests. Consequently, a larger number of receivers and packet requests can be served simultaneously with primary packets in each IDNC transmission.

From the above expression, we can see that, in order to maximize the coding density in each step, the selected cliques should be able to both maximize the number of primary edges and minimize the primary vertex set size. The number of primary vertices is clearly minimized if we both target the maximum number of receivers whenever possible and avoid ignoring any receiver that can be targeted along with any pre-selected set of receivers. Consequently, our selection of cliques should be only from the set of maximal cliques $K_m$, which cannot be a sub-clique in any larger cliques. The question now is which maximal clique maximizes the numerator. In other word, we need to find the best packet and receiver selection strategies so that the edge set size is maximized. To do so, we first need to derive an expression for the primary edge set size evolution after any arbitrary transmission. This will be the target of the following section.
4.3 Exact Primary Graph Evolution

To identify the packet and receiver strategies maximizing the edge set size in the IDNC graph, we need to derive expressions for this evolution after a transmission based on an arbitrary maximal clique $\kappa$. We thus first need to derive an expression for the edge set size of any IDNC graph, given its feedback matrix. One well-known approach to derive an expression of the edge set size of a graph is to first derive expressions for the degrees of its vertices.

4.3.1 Vertex Degrees

The degree of a vertex is defined as the number of edges adjacent to it. Since edges represent coding opportunities in the IDNC graph and since no coding opportunities exists between the vertices of each receiver, the vertex degree is thus equal to the number of coding opportunities this vertex has with vertices of other receivers. We define the primary degree of a vertex as the number of primary edges adjacent to this vertex and vertices in the primary graph. The following theorem presents an expression for the primary degree of an arbitrary vertex $v_{ij}$ given the state feedback matrix.

**Theorem 4.1.** The primary degree of a vertex $v_{ij}$ for an arbitrary state feedback matrix can be expressed as:

$$
\Delta_{ij} = \sum_{k=1}^{M} \left[ I_{j \in W_k} + I_{j \in H_k} (|W_k| - |W_k \cap L_i|) \right].
$$

where $I_x$ is an indicator function, which is equal to 1 if $x$ is true and 0 otherwise.

**Proof.** Consider an arbitrary vertex $v_{ij}$ in the graph. From the adjacency conditions C1 and C2 in Section 3.3.1, we can conclude the following facts:

- Vertex $v_{ij}$ is not adjacent to any primary vertex of the same receive $i$. 
• If \( j \in W_k \), \( v_{ij} \) cannot be adjacent to any primary vertex of receiver \( k \), due to violation of C2, except for vertex \( v_{kj} \) that satisfies C1. This contributes only one edge to its primary degree.

• If \( j \in H_k \), \( v_{ij} \) can be adjacent to any primary vertex of receiver \( k \) (induced from \( W_k \)), except for all vertices \( v_{kl} \) for which \( l \notin H_i \Rightarrow l \in L_i \). This contributes \(|W_k| - |W_k \cap L_i|\) edges to its primary degree.

From these facts, we can express the primary degree of a vertex \( v_{ij} \) as follows:

\[
\Delta_{ij} = \sum_{k=1}^{M} \left[ I_{j \in W_k} + I_{j \in H_k} (|W_k| - |W_k \cap L_i|) \right].
\]  
(4.3)

**Corollary 4.1.** In the broadcast scenarios, the degree of a vertex \( v_{ij} \) for an arbitrary state feedback matrix can be expressed as:

\[
\Delta_{ij} = \sum_{k=1}^{M} \left[ I_{j \in W_k} + I_{j \in H_k} (|W_k| - |W_k \cap W_i|) \right].
\]  
(4.4)

**Proof.** The corollary follows from the fact that \( W_i = L_i \) in the broadcast scenarios.  

From Theorem 4.1 and Corollary 4.1, we can see that, when \( j \in W_k \) or \( j \in L_k \), vertex \( v_{ij} \) is adjacent to only one vertex of \( k \), namely \( v_{kj} \), due to condition C1. It cannot be adjacent to any other vertices of \( k \) as this will violate condition C2. We say that this vertex \( v_{ij} \) is restricted by vertex \( v_{kj} \) and vise versa. The same fact applies to similar vertices in the secondary graph. All such restricted vertices are depicted in gray color in Figure 4.1. In this figure, each two vertices having the same packet index are adjacent to each other and are not adjacent to any other vertices. Indeed, if for example vertex \( v_{i3} \) is adjacent to any other vertex of \( k \) (for example \( v_{k6} \)), then it will allow a packet combination \( 3 \oplus 6 \), which is not instantly decodable for \( k \) and thus cannot be accepted.
On the other hand, we call the vertex $v_{ij}$, such that $j \in H_k$, as an unrestricted vertex with respect to $k$. All such vertices in the primary and secondary graphs are depicted in white color in Figure 4.1. This vertex $v_{ij}$ is adjacent to all vertices of $k$ in the primary and secondary graphs except for the vertices representing packets lacked by $i$, depicted in gray color. Otherwise, non-instantly decodable combinations will arise, such as $3 \oplus 5$ for example if we make $v_{i5}$ adjacent to $v_{k3}$. The number of these vertices in the primary graph is equal to $|W_k| - |W_k \cap L_i|$, as shown in Figure 4.1. This number is usually greater than one in most stages of the recovery phase. Consequently, when two vertices represent the loss of the same packet, they are restricted from being adjacent to a larger number of vertices of the other receiver.

### 4.3.2 Primary Edge Set Size

Using the primary vertex degree expressions derived in the previous section, we can now derive expressions for the primary edge set size using the well-known handshaking
lemma, stating that the edge set size of any graph is equal to half the sum of its vertex degrees \[78\]. The following theorem introduces an expression for the primary edge set size given any state feedback matrix.

**Theorem 4.2.** The primary edge set size for an arbitrary SFM can be expressed as:

\[
|E_\rho| = \frac{1}{2} \sum_{i=1}^{M} \left\{ \sum_{k=1}^{M} \left[ \psi_{ik} + (\psi_i - \varphi_{ik}) (\psi_k - \varphi_{ki}) \right] \right\},
\]

where \( \psi_{ik} = |W_i \cap W_k| \), \( \varphi_{ik} = |W_i \cap L_k| \) and \( \varphi_{ki} = |W_k \cap L_i| \)

**Proof.** To find the sum of all primary vertex degrees, we first define the sum \( \Sigma \Delta_i \) of all the primary degrees of the vertices induced by receiver \( i \), which can be expressed as:

\[
\Sigma \Delta_i = \sum_{j \in W_i} \left\{ \sum_{k=1, k \neq i}^{M} \left[ I_{j \in W_k} + I_{j \in H_k} \left( |W_k| - |W_k \cap L_i| \right) \right] \right\}
\]

\[
= \sum_{k=1, k \neq i}^{M} \left[ \left( \sum_{j \in W_i} I_{j \in W_k} \right) + \left( \sum_{j \in W_i} I_{j \in H_k} \right) \cdot (\psi_k - |W_k \cap L_i|) \right]
\]

\[
= \sum_{k=1, k \neq i}^{M} \left[ |W_i \cap W_k| + |W_i \cap H_k| \cdot (\psi_k - |W_k \cap L_i|) \right].
\]

We can easily infer that \( W_i \cap H_k = W_i \setminus (W_i \cap L_k) \). Consequently, we get:

\[
|W_i \cap H_k| = |W_i| - |W_i \cap L_k|.
\]

Substituting (4.7) in (4.6), summing over all \( i \), we get:

\[
|E_\rho| = \frac{1}{2} \sum_{i=1}^{M} \Sigma \Delta_i = \frac{1}{2} \sum_{i=1}^{M} \left\{ \sum_{k=1, k \neq i}^{M} \left[ \psi_{ik} + (\psi_i - \varphi_{ik}) (\psi_k - \varphi_{ki}) \right] \right\}.
\]
**Corollary 4.2.** In the broadcast scenarios, the edge set size for an arbitrary SFM can be expressed as:

\[
|E| = \frac{1}{2} \sum_{i=1}^{M} \left\{ \sum_{k=1}^{M} \left[ \psi_i \psi_k - (\psi_i + \psi_k - 1) \psi_{ik} + \psi_{ik}^2 \right] \right\}.
\]  

(4.9)

**Proof.** In the broadcast scenario, \(\psi_{ik} = \varphi_{ik} = \varphi_{ki}\). Substituting in (4.5), we get:

\[
|E| = \frac{1}{2} \sum_{i=1}^{M} \left\{ \sum_{k=1 \atop k \neq i}^{M} \left[ \psi_i \psi_k + (\psi_i - \psi_{ik})(\psi_k - \psi_{ik}) \right] \right\}
\[
= \frac{1}{2} \sum_{i=1}^{M} \left\{ \sum_{k=1 \atop k \neq i}^{M} \left[ \psi_i \psi_k - (\psi_i + \psi_k - 1) \psi_{ik} + \psi_{ik}^2 \right] \right\}.
\]  

(4.10)

\[\Box\]

The proof of Theorem 4.2 can be envisioned in a different way. Let \(Y_{ik}\) be the number of pairwise primary edges between receivers \(i\) and \(k\), which is depicted in Figure 4.1. We can see from the figure that the restricted primary vertices contribute to \(Y_{ik}\) by their number per receiver (i.e. \(|W_k \cap W_i| = \psi_{ik}\)). We can also observe that all mutually unrestricted primary and secondary vertices are connected to each other as a full bipartite subgraph. For the primary graph, these mutually unrestricted vertices contribute to \(Y_{ik}\) by \((|W_i| - |W_i \cap L_k|) (|W_k| - |W_k \cap L_i|) = (\psi_i - \varphi_{ik})(\psi_k - \varphi_{ki})\) edges. The final expression in (4.5) results from summing all \(Y_{ik}\) over all \(i\) and \(k\) and dividing by half to remove repetitions.

### 4.3.3 Exact Graph Evolution

Let \(\kappa\) be a chosen clique in the IDNC graph. Also define \(P_i\) as the source packet that is instantly decodable for receiver \(i \in T(\kappa)\). Moreover, let \(X_i\) be the reception indicator
function of receiver \(i\), which is equal to 1 if receiver \(i\) successfully receives the packet and zero otherwise. Finally, define \(\theta_{ik} = \psi_i - \varphi_{ik}\) and \(\theta_{ki} = \psi_k - \varphi_{ki}\). Using these definitions and the result of Theorem 4.2, we can introduce the expressions for the primary edge set size evolution, after any arbitrary transmission using the clique \(\kappa\), in the following theorem.

**Theorem 4.3.** For an arbitrary attempted clique \(\kappa\) at time \(t\), with a set of targeted receivers \(T(\kappa)\), the edge set size at time \(t + 1\) after this attempt can be expressed as:

\[
|E^{(t+1)}| = |E^{(t)}| + \frac{1}{2} \sum_{i \notin T(\kappa)} \left\{ \sum_{k \notin T(\kappa)} X_k (\theta_{ki} - 1) - \sum_{k \notin T(\kappa)} X_k \theta_{ik} + \sum_{k \notin T(\kappa)} X_k \theta_{ki} \right\} \\
+ \frac{1}{2} \sum_{i \in T(\kappa)} \left\{ \sum_{k \notin T(\kappa)} X_i \theta_{ik} - \sum_{k \notin T(\kappa)} X_k \theta_{ik} \right\} \\
+ \frac{1}{2} \sum_{i \in T(\kappa)} \left\{ \sum_{k \notin T(\kappa)} X_i (\theta_{ik} - 1) - \sum_{k \notin T(\kappa)} X_i \theta_{ki} - \sum_{k \notin T(\kappa)} X_i \theta_{ki} \\
+ \sum_{k \notin T(\kappa)} \left( X_i X_k - X_i \theta_{ki} - X_k \theta_{ik} \right) \\
+ \sum_{k \notin T(\kappa)} \left( X_i (\theta_{ik} - 1) + X_k (\theta_{ki} - 1) \\
- X_i X_k (\theta_{ik} + \theta_{ki} - 2 - X_i X_k) \right) \right\}. \quad (4.11)
\]

**Proof.** When an arbitrary clique \(\kappa\), with a given set of targeted receivers \(T(\kappa)\), is chosen for transmission, the values of \(\psi_i, \psi_k, \psi_{ik}, \varphi_{ik}\) and \(\varphi_{ki}\) in (4.5) change according to the cases depicted in Table 4.1 for each pair of receivers. In this table, \(X_{ik}\) is a joint reception...
<table>
<thead>
<tr>
<th>Cases</th>
<th>$\psi_i$</th>
<th>$\psi_k$</th>
<th>$\psi_{ik}$</th>
<th>$\varphi_{ik}$</th>
<th>$\varphi_{ki}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i \neq k$</td>
<td>$P_i/P_k$</td>
<td>$\psi_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_k \not\in T_i$, $P_k \not\in L_i$</td>
<td>$\psi_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_p$</td>
<td>$P_k \not\in W_i$, $P_k \not\in L_i$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_k \not\in W_i$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik} - X_k$</td>
</tr>
<tr>
<td>$i \in T_p$</td>
<td>$P_i \not\in L_k$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_i \not\in W_k$, $P_i \in L_k$</td>
<td>$\psi_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_p$</td>
<td>$P_i \not\in W_k$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \neq k$</td>
<td>$P_i \not\in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_i \not\in W_k$, $P_i \in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik} - X_i$</td>
<td>$\varphi_{ik} - X_i$</td>
</tr>
<tr>
<td>$i \in T_p$</td>
<td>$P_i \not\in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_i \not\in W_k$, $P_i \in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik} - X_i$</td>
<td>$\varphi_{ik} - X_i$</td>
</tr>
<tr>
<td>$i \in T_p$</td>
<td>$P_i \not\in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td>$\varphi_{ik}$</td>
</tr>
<tr>
<td>$i \in T_\sigma$</td>
<td>$P_i \not\in W_k$, $P_i \in L_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik} - X_k$</td>
<td>$\varphi_{ik} - X_k$</td>
</tr>
</tbody>
</table>

Table 4.1: Evolution Cases for Each Pair of Receivers

indicator function, which is equal to 1 if either $i$ or $k$ receives the transmitted packet and is zero otherwise. It is easy to see that $X_{ik}$ can be expressed as $X_{ik} = X_i + X_k - X_i X_k$. Let $Y_{ik}^{(t)}$ be the number of pairwise edges between the primary vertices of receivers $i$ and $k$ at time $t$ (i.e. $Y_{ik}^{(t)} = \psi_{ik} + (\psi_i - \varphi_{ik})(\psi_k - \varphi_{ki})$). According to the different targeting possibilities of receivers $i$ and $k$ illustrated in Table 4.1, the double summation in (4.11) can be divided so that we have one separate double summation per possibility, since all of these possibilities are mutually exclusive. In the double summation of a given possibility, the values of $\psi_i$, $\psi_k$, $\psi_{ik}$, $\varphi_{ik}$ and $\varphi_{ki}$ in (4.5) are replaced in the $Y_{ik}^{(t)}$ expression by their corresponding evolved values, depicted in the table, to obtain $Y_{ik}^{(t+1)}$. Applying these
changes, and using the definition $X_{ik} = X_i + X_k - X_i X_k$, we get the results in the list below for each of the possibilities. For some entries of the list, we will use explanatory figures for the evolution of $Y_{ik}^{(t)}$ to $Y_{ik}^{(t+1)}$ on the pairwise subgraph of Figure 4.1. For ease of illustration in these figures, receiver indices are removed from the vertices and all the preserved edges from $Y_{ik}^{(t)}$ to $Y_{ik}^{(t+1)}$ are also removed. The served vertices are marked in black and the added (removed) primary edges from $Y_{ik}^{(t)}$ to $Y_{ik}^{(t+1)}$ are represented by solid (dashed) lines. The added (removed) non-primary edges are represented with dash-dot (dotted) lines.

- For $i \notin T(\kappa), k \notin T(\kappa)$, there will be no change in $Y_{ik}^{(t)}$.

- For $i \notin T(\kappa), k \in T_\sigma(\kappa)$, if $P_k \notin L_i$ or if $P_k \notin W_i, P_k \in L_i$, we get:

$$Y_{ik}^{(t+1)} = Y_{ik}^{(t)}.$$

(4.12)

**Explanation 1**: Vertex $v_{iP_k}$ is either not existing (if $P_k \notin L_i$) or exists in the secondary graph (if $P_k \notin W_i, P_k \in L_i$). If $k$ receives, there may be some resulting edge changes at the vertices of $i$ in the secondary and/or complete graphs, but will not reflect in the primary graph. Thus, there will be no change in $Y_{ik}^{(t)}$.

- For $i \notin T(\kappa), k \in T_\sigma(\kappa)$ and $P_k \in W_i$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - \varphi_{ik} + X_k)(\psi_k - \varphi_{ki})$$

$$= Y_{ik}^{(t)} + X_k \theta_{ki}.$$  

(4.13)

**Explanation 2** (illustrated in Figure 4.2): If $k$ receives $P_k$ (packet 4 in fig.), the vertex $v_{iP_k}$ ($v_{i4}$ in fig.) will not be restricted by $v_{kP_k}$ ($v_{k4}$ in fig.) and thus becomes adjacent to the primary (and secondary) vertices of $k$ that are not restricted by $i$’s vertices ($v_{k6}$, $v_{k7}$, $v_{k11}$ in fig.). The disappearance of the common edge between $v_{iP_k}$ and $v_{kP_k}$ does not affect $Y_{ik}^{(t+1)}$ as it is not a primary edge at $Y_{ik}^{(t)}$. In the figure,
the number of increased primary edges is 2 (which is equal to $\theta_{ki}$)

- For $i \notin \mathcal{T}(\kappa)$, $k \in \mathcal{T}_p(\kappa)$ and $P_k \notin \mathcal{L}_i$, we get:

\[
Y^{(t+1)}_{ik} = \psi_{ik} + (\psi_i - \varphi_{ik})(\psi_k - X_k - \varphi_{ki})
\]

\[
= Y^{(t)}_{ik} - X_k \theta_{ik} .
\] (4.14)

**Explanation 3** (illustrated in Figure 4.3): If $k$ receives (packet 6 in fig.), an unrestricted $v_{kP_k}$ ($v_{k6}$ in fig.) with respect to $i$ disappears with all its edges, including its edges to the primary (and secondary) vertices of $i$ except those restricted by $k$ (and thus are not connected to $v_{kP_k}$ in the first place). The number of these primary edges is equal to $\theta_{ik}$ (3 in fig.). Since this vertex $v_{kP_k}$ is unrestricted with respect to $i$, its disappearance will not add any edges between the primary vertices of $i$ and $k$ (i.e. no additional coding opportunities are created).

- For $i \notin \mathcal{T}(\kappa)$, $k \in \mathcal{T}_p(\kappa)$ and $P_k \notin \mathcal{W}_i, P_k \in \mathcal{L}_i$, we get:

\[
Y^{(t+1)}_{ik} = \psi_{ik} + (\psi_i - \varphi_{ik})(\psi_k - X_k - \varphi_{ki} + X_k)
\]

\[
= Y^{(t)}_{ik} .
\] (4.15)
Explanation 4: If $k$ receives, vertex $v_{iP_k}$ will no longer be restricted by $v_{kP_k}$ and will be connected to other vertices of $k$. But since vertex $v_{iP_k}$ is located in the secondary graph, all of these edges will not be part of the primary graph and thus we do not see their effect in $Y_{ik}^{(t+1)}$.

- For $i \not\in T(\kappa)$, $k \in T_p(\kappa)$ and $P_k \in \mathcal{W}_i$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} - X_k + (\psi_i - \varphi_{ik} + X_k)(\psi_k - X_k - \varphi_{ki} + X_k)$$

$$= Y_{ik}^{(t)} + X_k(\theta_{ki} - 1) \quad (4.16)$$

Explanation 5 (illustrated in Figure 4.4): If $k$ receives (packet 3 in fig.), vertex $v_{iP_k}$ ($v_{i3}$ in fig.), which is in the primary graph (since $P_k \in \mathcal{W}_i$), will no longer be restricted by $v_{kP_k}$ ($v_{3k}$ in fig.) and thus will be adjacent to all vertices of $k$ in the primary (and secondary) graph ($v_{6k}, v_{8k}, v_{11k}$ in fig.) except those restricted by other vertices of $i$ ($v_{1k}, v_{2k}, v_{4k}$ in fig.). The number of these primary vertices is equal to $\theta_{ki}$ (2 in fig.). The term $-1$ in the bracket represents the disappearance of the edge connecting $v_{iP_k}$ and $v_{kP_k}$ in the primary graph since $v_{kP_k}$ disappeared.
The explanation for this case is same as Explanation 1 when $i$ and $k$ are swapped.

- For $i \in T_\sigma(\kappa)$, $k \notin T(\kappa)$, if $P_i \notin L_k$ or $P_i \notin W_k$, $P_i \in L_k$, we get:

$$Y^{(t+1)}_{ik} = Y^{(t)}_{ik}. \quad (4.17)$$

The explanation for this case is the same as Explanation 2 when $i$ and $k$ are swapped.

- For $i \in T_\sigma(\kappa)$, $k \notin T(\kappa)$, if $P_i \in W_k$

$$Y^{(t+1)}_{ik} = \psi_{ik} + (\psi_i - \varphi_{ik})(\psi_k - \varphi_{ki} + X_i)$$
$$= Y^{(t)}_{ik} + X_i \theta_{ik}. \quad (4.18)$$

The explanation for this case is the same as Explanation 2 when $i$ and $k$ are swapped.

- For $i \in T_\sigma(\kappa)$, $k \in T_\sigma(\kappa)$, if $P_i \notin L_k$ or $P_i \notin W_k$, $P_i \in L_k$, we get:

$$Y^{(t+1)}_{ik} = Y^{(t)}_{ik}. \quad (4.19)$$

**Explanation 6:** Both targeted vertices are in the secondary graph and are not restricting any vertices in the primary graph. Thus, if either or both receivers
receive, no edges will be added between their vertices in the primary graph.

- The case \( i \in T_{\sigma}(\kappa), k \in T_{\sigma}(\kappa) \) and \( P_i \in W_k \), is an invalid case.

   **Explanation 7:** If \( P_i \in W_k \), then \( v_{iP_i} \) is restricted by a vertex \( v_{kP_i} \) in the primary graph and thus cannot be served with another vertex of \( k \) from the secondary graph.

- For \( i \in T_{\sigma}(\kappa), k \in T_{\rho}(\kappa) \), and \( P_i \notin L_k \), we get:

\[
Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - \varphi_{ik}) \left( \psi_k - X_k - \varphi_{ki} \right) \\
= Y_{ik}^{(t)} - X_k \theta_{ik} .
\]  

(4.20)

The explanation for this case is similar to Explanation 3.

- The case \( i \in T_{\sigma}(\kappa), k \in T_{\rho}(\kappa) \) and \( P_i \notin W_k, P_i \in L_k \), is an invalid case.

   **Explanation 8:** If \( P_i \in L_k \) but not \( W_k \), then \( v_{iP_i} \) will be restricted by a vertex \( v_{kP_i} \) in the secondary graph and thus cannot be served with another vertex of \( k \) from the primary graph.

- For \( i \in T_{\sigma}(\kappa), k \in T_{\rho}(\kappa) \), and \( P_i \in W_k \), we get:

\[
Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - \varphi_{ik}) \left( \psi_k - X_k - \varphi_{ki} + X_k \right) \\
= Y_{ik}^{(t)} .
\]  

(4.21)

The explanation of this case is similar to Explanation 4.

- For \( i \in T_{\rho}(\kappa), k \notin T(\kappa) \), and \( P_i \notin L_k \), we get:

\[
Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - \varphi_{ik}) \left( \psi_k - \varphi_{ki} \right) \\
= Y_{ik}^{(t)} - X_i \theta_{ki} .
\]  

(4.22)

The explanation for this case is similar to Explanation 3, by switching \( k \) with \( i \).
• For $i \in T_{\rho}(\kappa)$, $k \notin T(\kappa)$, and $P_i \notin W_k, P_i \in L_k$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - X_i - \varphi_{ik} + X_i) (\psi_k - \varphi_{ki})$$
$$= Y_{ik}^{(t)} . \quad (4.23)$$

The explanation of this case is the same as Explanation 4 when $i$ and $k$ are swapped.

• For $i \in T_{\rho}(\kappa)$, $k \notin T(\kappa)$, and $P_i \in W_k$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} - X_i + (\psi_i - X_i - \varphi_{ik} + X_i) (\psi_k - \varphi_{ki} + X_i)$$
$$= Y_{ik}^{(t)} + X_i (\theta_{ik} - 1) . \quad (4.24)$$

The explanation of this case is the same as Explanation 5 when $i$ and $k$ are swapped.

• For $i \in T_{\rho}(\kappa)$, $k \in T_{\sigma}(\kappa)$, and $P_i \notin L_k$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - X_i - \varphi_{ik}) (\psi_k - \varphi_{ki})$$
$$= Y_{ik}^{(t)} - X_i \theta_{ki} . \quad (4.25)$$

The explanation of this case is the same as Explanation 3 when $i$ and $k$ are swapped.

• For $i \in T_{\rho}(\kappa)$, $k \in T_{\sigma}(\kappa)$, and $P_i \notin W_k, P_i \in L_k$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - X_i - \varphi_{ik} + X_i) (\psi_k - \varphi_{ki})$$
$$= Y_{ik}^{(t)} . \quad (4.26)$$

The explanation of this case is the same as Explanation 4 when $i$ and $k$ are swapped.

• The case $i \in T_{\rho}(\kappa), k \in T_{\sigma}(\kappa)$, and $P_i \in W_k$ is an invalid case. The explanation of this case is the same as Explanation 7.
For $i \in T_\rho(\kappa), k \in T_\rho(\kappa)$ and $P_k \notin \mathcal{L}_i$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} + (\psi_i - X_i - \varphi_{ik}) (\psi_k - X_k - \varphi_{ki})$$

$$= Y_{ik}^{(t)} + X_i X_k - X_i \theta_{ki} - X_k \theta_{ik} .$$

(4.27)

**Explanation 9**: If one receiver receives (i.e. $X_i X_k = 0$), we get the same situation explained in Explanation 3. If both receivers receive (illustrated in Figure 4.5), both unrestricted vertices ($v_{i7}, v_{k6}$ in fig.) will disappear with their edges including their primary edges to each others’ mutual unrestricted vertices ($v_{k8}$ for $i$ and $v_{i5}, v_{i9}$ for $k$). In this case, the term $X_i X_k = 1$ compensates for the removal of their common edge twice in the other two terms in (4.27).

- The case $i \in T_\rho(\kappa), k \in T_\rho(\kappa)$, and $P_i \notin \mathcal{W}_k, P_i \in \mathcal{L}_k$ is an invalid case. The explanation of this case is the same as Explanation 8.

- For $i \in T_\rho(\kappa), k \in T_\rho(\kappa)$ and $P_k \in \mathcal{W}_i$, we get:

$$Y_{ik}^{(t+1)} = \psi_{ik} - X_{ik} + (\psi_i - X_i - \varphi_{ik} + X_{ik}) (\psi_k - X_k - \varphi_{ki} + X_{ik})$$

$$= Y_{ik}^{(t)} + X_i (\theta_{ik} - 1) + X_k (\theta_{ki} - 1) - X_i X_k (\theta_{ik} + \theta_{ki} - 2 - X_i X_k) .$$

(4.28)
Explanation 10: If one of the receivers receives (i.e. $X_i X_k = 0$), we get the case explained in Explanation 5 for the other receiver. If both receivers receive (i.e. $X_i X_k = 1$), only the served vertices and their common edge will disappear from the primary graph ($v_{i3}$ and $v_{k3}$ and their mutual edge as illustrated in Figure 4.6).

The theorem follows by substituting the above equations in (4.5), and knowing that the summations of $Y_{ik}(t+1)$ and $Y_{ik}(t)$ over all cases and all receiver pairs are equal to $|\mathcal{E}(t+1)|$ and $|\mathcal{E}(t)|$, respectively.

From Corollary 4.2, we can introduce the following corollary about the edge set size evolution in the broadcast scenarios. Note that, for the broadcast case, $\varphi_{ik} = \varphi_{ki} = \psi_{ik}$, and thus $\theta_{ik} = \psi_{i} - \psi_{ik}$ and $\theta_{ki} = \psi_{k} - \psi_{ik}$.

Corollary 4.3. In the broadcast scenarios, the edge set size at time $t+1$, after attempting
Chapter 4. IDNC Graph Evolution

<table>
<thead>
<tr>
<th>Cases</th>
<th>$i$</th>
<th>$k$</th>
<th>$P_i/P_k$</th>
<th>$\psi_i$</th>
<th>$\psi_k$</th>
<th>$\psi_{ik}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\notin T(\kappa))</td>
<td>(\notin T(\kappa))</td>
<td>-</td>
<td>$\psi_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td></td>
</tr>
<tr>
<td>(\notin T(\kappa))</td>
<td>(\in T(\kappa))</td>
<td>$P_k \notin W_i$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td></td>
</tr>
<tr>
<td>(\notin T(\kappa))</td>
<td>(\in T(\kappa))</td>
<td>$P_k \in W_i$</td>
<td>$\psi_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik} - X_k$</td>
<td></td>
</tr>
<tr>
<td>(\in T(\kappa))</td>
<td>(\notin T(\kappa))</td>
<td>$P_i \notin W_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik}$</td>
<td></td>
</tr>
<tr>
<td>(\in T(\kappa))</td>
<td>(\notin T(\kappa))</td>
<td>$P_i \in W_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k$</td>
<td>$\psi_{ik} - X_i$</td>
<td></td>
</tr>
<tr>
<td>(\in T(\kappa))</td>
<td>(\in T(\kappa))</td>
<td>$P_i \notin W_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik}$</td>
<td></td>
</tr>
<tr>
<td>(\in T(\kappa))</td>
<td>(\in T(\kappa))</td>
<td>$P_i \in W_k$</td>
<td>$\psi_i - X_i$</td>
<td>$\psi_k - X_k$</td>
<td>$\psi_{ik} - X_{ik}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Evolution Cases for Each Pair of Receivers in the broadcast scenario

An arbitrary clique \(\kappa\) at time \(t\), with a set of targeted receivers \(T(\kappa)\), can be expressed as:

\[
|\mathcal{E}^{(t+1)}| = |\mathcal{E}^{(t)}| + \frac{1}{2} \sum_{i \notin T(\kappa)} \left\{ \sum_{k \in T(\kappa)} X_k (\theta_{ik} - 1) - \sum_{k \notin T(\kappa)} X_k \theta_{ik} \right\} \\
+ \frac{1}{2} \sum_{i \in T(\kappa)} \left\{ \sum_{k \notin T(\kappa)} X_i (\theta_{ik} - 1) - \sum_{k \notin T(\kappa)} X_i \theta_{ki} \right\} \\
+ \sum_{k \in T(\kappa)} X_i X_k - X_i \theta_{ki} - X_k \theta_{ik} \\
+ \sum_{k \in T(\kappa)} \left( X_i (\theta_{ik} - 1) + X_k (\theta_{ki} - 1) \\
- X_i X_k (\theta_{ik} + \theta_{ki} - 2 - X_i X_k) \right) \right\} 
\] (4.29)

Proof. The proof follows the same method of the proof of Theorem 4.3 using Table 4.2.

Using the formulae (4.11) and (4.29) in Theorem 4.3 and Corollary 4.3, we investigate the packet selection strategy maximizing the edge set size evolution in the next section.
4.4 Packet Selection Strategy

From (4.11), we can draw the following observations for each pair of receivers.

**Case 1:** $i \notin \mathcal{T}(\kappa)$

- When $k$ is targeted by a primary or secondary packet wanted by $i$, the number of its pairwise primary edges with $i$'s vertices (i.e. $Y_{ik}^{(t+1)}$) is either increased (if $X_k = 1$) or unchanged (if $X_k = 0$).

- When $k$ is targeted by a packet that is not lacked (i.e. received) by $i$, $Y_{ik}^{(t+1)}$ is either decreased (if $X_k = 1$ and $k \in \mathcal{T}_\rho(\kappa)$) or unchanged (if $X_k = 0$ or $k \in \mathcal{T}_\sigma(\kappa)$).

**Case 2:** $i \in \mathcal{T}_\sigma(\kappa)$

- When the packet targeting $i$ is wanted by the un-targeted receiver $k$, $Y_{ik}^{(t+1)}$ is either increased (if $X_i = 1$) or unchanged (if $X_i = 0$).

- When the packet targeting $i$ is received by the targeted receiver $k$, $Y_{ik}^{(t+1)}$ is either decreased (if $X_k = 1$ and $k \in \mathcal{T}_\rho(\kappa)$) or unchanged (if $X_k = 0$ or $k \in \mathcal{T}_\sigma(\kappa)$).

**Case 3:** $i \in \mathcal{T}_\rho(\kappa)$ and $k \notin \mathcal{T}(\kappa)$ or $k \in \mathcal{T}_\sigma(\kappa)$

- When the packet targeting $i$ is wanted by the un-targeted receiver $k$, $Y_{ik}^{(t+1)}$ is increased (if $X_i = 1$) or unchanged (if $X_i = 0$).

- When the packet targeting $i$ is received by the un-targeted receiver $k$, $Y_{ik}^{(t+1)}$ is decreased (if $X_i = 1$) or unchanged (if $X_i = 0$).

- When the packet targeting $i$ is received by the targeted receiver $k$, $Y_{ik}^{(t+1)}$ is decreased (if $X_k = 1$ and $k \in \mathcal{T}_\sigma(\kappa)$) or unchanged (if $X_k = 0$).

**Case 4:** $i \in \mathcal{T}_\rho(\kappa)$ and $k \in \mathcal{T}_\rho(\kappa)$

- Possibility 1: $X_i = 0$ and $X_k = 0$
- \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} \) whether \( P_i \) is in \( \mathcal{W}_k \) or not in \( \mathcal{L}_k \).

• Possibility 2: \( X_i = 1 \) and \( X_k = 0 \)
  - For \( P_i \notin \mathcal{L}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} - (\psi_k - \varphi_{ki}) \).
  - For \( P_i \in \mathcal{W}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} + (\psi_k - \psi_{ki} - 1) \).

• Possibility 3: \( X_i = 0 \) and \( X_k = 1 \)
  - For \( P_i \notin \mathcal{L}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} - (\psi_i - \varphi_{ik}) \).
  - For \( P_i \in \mathcal{W}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} + (\psi_i - \varphi_{ik} - 1) \).

• Possibility 4: \( X_i = 1 \) and \( X_k = 1 \)
  - For \( P_i \notin \mathcal{L}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} + 1 - (\psi_k - \varphi_{ki}) - (\psi_i - \varphi_{ik}) \leq Y_{ik}^{(t)} - 1 \).
  - For \( P_i \in \mathcal{W}_k \), \( Y_{ik}^{(t+1)} = Y_{ik}^{(t)} - 1 \).

The last inequality in Possibility 4 for \( P_i \notin \mathcal{L}_k \) arises from two facts. First, packet \( P_i \) is in \( \mathcal{W}_i \) but not \( \mathcal{L}_k \) and thus cannot be part of their intersection. Second, packet \( P_k \), which is in \( \mathcal{W}_k \), cannot be in \( \mathcal{L}_i \). Indeed, if \( P_k \) was in \( \mathcal{L}_i \), it would have been restricted by a vertex \( v_{iP_k} \) and could have not been served with \( v_{iP_k} \). Thus, \( \psi_i \geq \varphi_{ik} + 1 \) and \( \psi_k \geq \varphi_{ki} + 1 \) and thus the left hand term will always be less than or equal to \( Y_{ik}^{(t)} - 1 \).

From all the above cases, we can see that serving a common wanted packet for \( i \) and \( k \) always results in a larger increase or smaller decrease in the number of their pairwise primary edges, whether one or both receivers are targeted by this packet. Generalizing this result to all pairs, we can conclude that serving the maximum number of packets with large Demands set sizes tends to maximize the resulting number of coding opportunities.

However, this finding does not mean that the coding density will be increased at all stages of the recovery transmission phase. The success of this strategy to maximize the coding density all the time is conditioned on the continuous presence of packets with large Demands sets. In general, such packets can be found in the beginning of the recovery
transmission phase. In this case, serving these packets both eliminates the large number of vertices requesting them and increases the number of edges in the resulting graph, which greatly improves the coding density. At the same time, the Demands sets, with initial large sizes, will considerably decrease after each transmission, which may end up with graphs having no packets with large Demands sets. In this case, this packet selection strategy will result in serving a small number of vertices in each transmission, which may decrease the coding density. We will illustrate and analyze these effects in Section 4.7.

4.5 Graph Evolution Given Set Cardinalities

The exact evolution expressions of coding opportunities in (4.11) and (4.29) greatly depend on the packets selected for each transmission, and thus were useful to draw some insights on the packet selection strategy maximizing the coding density. However, this great dependency of (4.11) and (4.29) on the selected packets makes it difficult to infer from them any receiver selection strategies maximizing the coding density. Nonetheless, selecting targeted receivers is sometimes more crucial in optimizing several parameters, such as the decoding delay and most importantly the completion delay (as will be discussed in Chapter 5). Consequently, it is very important to find another approach to express the edge set size evolution in a way that can clarify the best receiver selection strategy improving it.

One of these approaches may be expressing the edge set size evolution, after ignoring the content of the different sets and focusing on their cardinalities. In this case, the contents of the Has, Lacks and Wants sets of receiver \( i \) become random sets of packets of sizes \( \varrho_i, \varphi_i \) and \( \psi_i \), respectively, all drawn from the pool of the \( N \) original source packets. Consequently, the indicator functions \( I_{j\in W_k} \) and \( I_{j\in H_k} \) become uniformly distributed random variables with mean \( \frac{\psi_k}{N} \) and \( \frac{\varphi_k}{N} \), respectively. Moreover, the set \( W_i \cap W_k \) becomes the intersection of two random sets drawn from the same pool and of known sizes \( \psi_i \) and
ψ_k. Consequently, ψ_{ik} becomes a random variable following a hypergeometric distribution. The same applies to ϕ_{ik} and ϕ_{ki}. Thus, the derived expressions of the edge set size evolution would be an expectation over these random variables as well as the reception probabilities of the transmitted packet at the different receivers.

To derive these expressions for the multicast and broadcast scenarios, we will follow the same approach employed for the exact expressions, by first computing the expected primary vertex degrees and the expected edge set size, given the set cardinalities.

### 4.5.1 Expected Vertex Degrees

The following theorem introduces an expression for the expected primary degree of any vertex induced by receiver i, given the cardinalities of the different sets.

**Theorem 4.4.** For given φ, ψ and ϑ vectors, the expected primary degree of any of the vertices induced by receiver i is equal to:

\[
E[\Delta_i] = \sum_{k=1}^{M} \psi_k \left( \frac{1 + \varnothing_k \varnothing_i}{N - 1} \right).
\]  

(4.30)

**Proof.** We will start this proof from the exact vertex degree expression in (4.2). Ignoring the content of the different sets, we can derive the expression for the expected degree of a vertex of receiver i as follows:

\[
E[\Delta_i] = E[\Delta_{ij}] = \sum_{k=1}^{M} E[I_{j \in W_k}] + E[I_{j \in H_k}] |W_k| - E[I_{j \in H_k} \cdot |W_k \cap L_i|] \\
= \sum_{k=1}^{M} \frac{\psi_k}{N} + \frac{\varnothing_k \psi_k}{N} - E[I_{j \in H_k} \cdot |W_k \cap L_i|].
\]  

(4.31)

Note that the indicator function in the last expectation term can be only zero or one. Consequently, the expectation of its multiplication with |W_k \cap L_i| can be only evaluated for I_{j \in H_k} = 1. If this is the case, packet j cannot be in the intersection of W_k and L_i,
since it is not in $W_k$. But since $j \in W_i$ (or else it would not have had a vertex in the primary graph), and thus $j \in L_i$, the elements of the random set $W_k \cap L_i$ can only be drawn from the remaining $\varphi_i - 1$ packets in $L_i$, and from the set of remaining $N - 1$ source packets. Consequently, the random variable representing its cardinality is still a hypergeometric random variable with parameters $(N - 1, \psi_k, \varphi_i - 1)$. We can thus evaluate the expectation in (4.31) as:

$$
E \left[ I_{j \in H_k} \cdot |W_k \cap L_i| \right] = \sum_{n=1}^{N-1} n \ P \left[ |W_k \cap L_i| = n \right] \cdot \ P \left[ I_{j \in H_k} = 1 \right]
$$

$$
= \sum_{n=1}^{N-1} n \ P \left[ |W_k \cap L_i| = n \right] \cdot \sum_{k=1}^{N-1} \binom{\varphi_i - 1}{n} \binom{N - 1 - \varphi_i + 1}{\psi_k - n} \ \binom{\psi_k}{n} \binom{N - 1}{\psi_k - n}
$$

$$
= \frac{\psi_k (\varphi_i - 1)}{N(N - 1)}. \tag{4.32}
$$

Substituting (4.32) in (4.31), we get:

$$
E \left[ \Delta_i \right] = \sum_{k=1}^{M} \binom{\psi_i}{N} \sum_{k \neq i} \binom{\psi_k}{N} \left[ 1 + \varrho_k \left( 1 - \frac{\varphi_i - 1}{N - 1} \right) \right]
$$

$$
= \sum_{k=1}^{M} \binom{\psi_k}{N} \left( 1 + \frac{\varrho_k \psi_i}{N - 1} \right). \tag{4.33}
$$

Note that for the broadcast scenarios, the expected vertex degree expression is the same as (4.30) but in this case $\varrho_i + \psi_i = N$.

### 4.5.2 Expected Edge Set Size

The following theorem introduces the expression of the primary edge set size, after ignoring the content of the Has, Lacks and Wants sets and considering only their cardinalities.
Theorem 4.5. For given $\varrho$, $\varphi$ and $\psi$ vectors, the expected edge set cardinality of the primary graph is equal to:

$$E[|E_\rho|] = \frac{1}{2} \sum_{i=1}^{M} \psi_i \left\{ \sum_{k=1}^{M} \left( \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k \varrho_i}{N-1} \right) \right) \right\}$$

$$= \frac{1}{2} \sum_{i=1}^{M} \psi_i E[\Delta_i] . \quad (4.34)$$

Proof. We will start our proof from (4.6) in the proof of Theorem 4.2. Ignoring the content of the different sets, we can derive the expression for the expected primary edge set size as follows:

$$E[|E_\rho|] = \frac{1}{2} \sum_{i=1}^{M} E[\Sigma \Delta_i]$$

$$= \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} E[|W_i \cap W_k|] + \psi_k E \left[ \sum_{j \in W_i} I_{j \in H_k} \right] - E \left[ \sum_{j \in W_i} I_{j \in H_k} \cdot |W_k \cap L_i| \right]$$

$$= \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} \frac{\psi_i \psi_k}{N} + \frac{\psi_k \psi_i \varrho_k}{N} - \sum_{j \in W_i} E[I_{j \in H_k} \cdot |W_k \cap L_i|] . \quad (4.35)$$

Substituting (4.32) in (4.35), we get:

$$E[|E_\rho|] = \frac{1}{2} \sum_{i=1}^{M} \sum_{k=1}^{M} \frac{\psi_i \psi_k}{N} + \frac{\psi_k \psi_i \varrho_k}{N} - \frac{\psi_i \varrho_k \psi_k (\varphi_i - 1)}{N(N-1)}$$

$$= \frac{1}{2} \sum_{i=1}^{M} \psi_i \left\{ \sum_{k=1}^{M} \frac{\psi_k}{N} \left[ 1 + \varrho_k \left( 1 - \frac{\varphi_i - 1}{N-1} \right) \right] \right\}$$

$$= \frac{1}{2} \sum_{i=1}^{M} \psi_i \left\{ \sum_{k=1}^{M} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k \varphi_i}{N-1} \right) \right\}$$

$$= \frac{1}{2} \sum_{i=1}^{M} \psi_i E[\Delta_i] . \quad (4.36)$$
Note that for the broadcast scenarios, the expression for the expected primary edge set size is the same as (4.34) but in this case $\varrho_i + \psi_i = N$.

### 4.5.3 Expected Graph Evolution

Since the expression of the expected edge set size in (4.34) is expressed as a weighted sum of the expected vertex degrees, we will first derive an expression for the expected primary degree evolution, after any arbitrary attempted clique $\kappa$, in the following theorem.

**Theorem 4.6.** For a given maximal clique $\kappa$, chosen for transmission at time $t$, the expected primary degree of a receiver $i$’s vertex at time $t + 1$ is expressed as:

$$
E \left[ \Delta_{i \in T(\kappa)}^{(t+1)} \right] = E \left[ \Delta_i^{(t)} \right] + \sum_{k=1, k \neq i}^M q_k \xi_k - \sum_{k \in T_{\rho}(\kappa)} \Phi_k(q_i) + \sum_{k \in T_{\sigma}(\kappa)} \Lambda_k(q_i) \tag{4.37}
$$

if $i \in T(\kappa)$ and is expressed as:

$$
E \left[ \Delta_{i \notin T(\kappa)}^{(t+1)} \right] = E \left[ \Delta_i^{(t)} \right] - \sum_{k \in T_{\rho}(\kappa)} \Phi_k(0) + \sum_{k \in T_{\sigma}(\kappa)} \Lambda_k(0) \tag{4.38}
$$

if $i \notin T(\kappa)$, such that:

$$
\Phi_k(z) = \frac{q_k}{N} \left( 1 + \frac{(q_k - \psi_k + 1)(\varrho_i + z)}{N - 1} \right) \tag{4.39}
$$

$$
\Lambda_k(z) = \frac{q_k \psi_k (\varrho_i + z)}{N(N - 1)} \tag{4.40}
$$

$$
\xi_k = \frac{\psi_k q_k}{N(N - 1)} \cdot \tag{4.41}
$$

**Proof.** When a clique $\kappa$ is chosen for transmission at time $t$, each member $k$ of the targeted receiver set $T(\kappa)$ may receive the coded packet with probability $q_k$. Let $X = [X_1, \ldots, X_M]$ be the vector of reception indicators. Using the result of Theorem 4.4, we can derive the
expression of the expected primary degree of receiver $i \in \mathcal{T}(\kappa)$ at time $t+1$, conditioned on the random vector $X$, as follows:

\[
\mathbb{E} \left[ \Delta_{i \in \mathcal{T}(\kappa)}^{(t+1)} \mid X \right] = \sum_{k \in \mathcal{T}_\kappa(k) \setminus k \neq i} \frac{\psi_k - X_k}{N} \left( 1 + \frac{(\varrho_k + X_k)(\varrho_i + X_i)}{N - 1} \right) \\
+ \sum_{k \in \mathcal{T}_\kappa(k) \setminus k \neq i} \frac{\psi_k}{N} \left( 1 + \frac{(\varrho_k + X_k)(\varrho_i + X_i)}{N - 1} \right) \\
+ \sum_{k \notin \mathcal{T}(\kappa)} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k (\varrho_i + X_i)}{N - 1} \right) \\
= \sum_{k=1}^{M} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k \varrho_i}{N - 1} \right) + \sum_{k=1}^{M} \frac{\psi_k \varrho_k X_i}{N(N - 1)} \\
- \sum_{k \in \mathcal{T}_\kappa(k) \setminus k \neq i} \frac{X_k}{N} \left( 1 + \frac{(\varrho_k - \psi_k + X_k)(\varrho_i + X_i)}{N - 1} \right) \\
+ \sum_{k \notin \mathcal{T}(\kappa) \setminus k \neq i} \frac{\psi_k X_k (\varrho_i + X_i)}{N(N - 1)}.
\]

(4.42)

The first term in (4.42) is obviously the expected vertex degree of receiver $i$ at time $t$. Since $X_i$ and $X_k$ are independent bernoulli random variables for all $i$ and $k$, and since $\mathbb{E} X \{X_k\} = \mathbb{E} X \{X_i^2\} = q_k$, we can derive the expected degree of receiver $i$ after serving the maximal clique $\kappa$ as follows:

\[
\mathbb{E} \left[ \Delta_{i \in \mathcal{T}(\kappa)}^{(t+1)} \right] = \mathbb{E} X \left\{ \mathbb{E} \left[ \Delta_{i \in \mathcal{T}(\kappa)}^{(t+1)} \mid X \right] \right\} \\
= \mathbb{E} \left[ \Delta_i^{(t)} \right] + \sum_{k=1}^{M} \psi_k \varrho_k \mathbb{E} X \{X_i\} \frac{1}{N(N - 1)} + \sum_{k \in \mathcal{T}_\kappa(k) \setminus k \neq i} \frac{\psi_k \mathbb{E} X \{X_k\} (\varrho_i + \mathbb{E} X \{X_i\})}{N(N - 1)} \\
- \sum_{k \in \mathcal{T}_\kappa(k) \setminus k \neq i} \frac{\mathbb{E} X \{X_k\}}{N} + \frac{\left( \mathbb{E} X \{X_k\} (\varrho_k - \psi_k) + \mathbb{E} X \{X_k^2\}\right)(\varrho_i + \mathbb{E} X \{X_i\})}{N(N - 1)}.
\]

(4.43)
Using the definitions of $\xi_i$, $\Phi_i(z)$ and $\Lambda_i(z)$ in (4.41), (4.39) and (4.40), respectively, in (4.43), we get:

$$
E \left[ \Delta^{(t+1)}_{i\in T(\kappa)} \right] = E \left[ \Delta^{(t)}_i \right] + \sum_{k=1}^{M} q_i \xi_k - \sum_{k\in T_\rho(\kappa) \ k\neq i} \Phi_k(q_i) + \sum_{k\in T_\sigma(\kappa) \ k\neq i} \Lambda_k(q_i). 
$$

(4.44)

The expression for $E \left[ \Delta^{(t+1)}_{i\notin T(\kappa)} \right]$ can be derived by setting $X_i = 0$ in (4.42) and continuing the derivation.

**Corollary 4.4.** For a given maximal clique $\kappa$, chosen for transmission at time $t$ in a broadcast scenario, the expected degree of a receiver $i$ vertex at time $t+1$ is expressed as:

$$
E \left[ \Delta^{(t+1)}_{i\in T(\kappa)} \right] = E \left[ \Delta^{(t)}_i \right] + \sum_{k=1}^{M} q_i \xi_k - \sum_{k\in T(\kappa) \ k\neq i} \Phi_k(q_i) 
$$

(4.45)

if $i \in T$ and is expressed as:

$$
E \left[ \Delta^{(t+1)}_{i\notin T} \right] = E \left[ \Delta^{(t)}_i \right] - \sum_{k\in T} \Phi_k(0) 
$$

(4.46)

if $i \notin T$.

**Proof.** The corollary follows by removing the summation term in $\sum_{k\in T_\sigma(\kappa)}$ representing $k \in T_\sigma(\kappa)$ being in the secondary graph, and continuing the derivation.

Having the expressions of the expected degree evolution derived, we derive the expression of the expected edge set size evolution in the following theorem.

**Theorem 4.7.** For a given maximal clique $\kappa$, chosen for transmission at time $t$, the expected edge set cardinality of the IDNC primary graph at time $t+1$ is expressed as:

$$
E \left[ |\mathcal{E}^{(t+1)}_\rho| \right] = E \left[ |\mathcal{E}^{(t)}_\rho| \right] - \frac{1}{2} \sum_{i\in T_\rho(\kappa)} q_i \left( E \left[ \Delta^{(t)}_i \right] + \gamma_i \right) + \frac{1}{2} \sum_{i\in T(\kappa)} \psi_i \alpha_i + \frac{1}{2} \sum_{i\notin T(\kappa)} \psi_i \beta_i, 
$$

(4.47)
where

\[
\alpha_i = \sum_{k=1 \atop k \neq i}^M q_i \xi_k - \sum_{k \in T_\rho(\kappa) \atop k \neq i} \Phi_k(q_i) + \sum_{k \in T_\sigma(\kappa) \atop k \neq i} \Lambda_k(q_i), \tag{4.48}
\]

\[
\beta_i = -\sum_{k \in T_\rho(\kappa)} \Phi_k(0) + \sum_{k \in T_\sigma(\kappa)} \Lambda_k(0), \tag{4.49}
\]

\[
\gamma_i = \sum_{k=1 \atop k \neq i}^M \xi_k - \sum_{k \in T_\rho(\kappa) \atop k \neq i} \Phi_k(1) + \sum_{k \in T_\sigma(\kappa) \atop k \neq i} \Lambda_k(1). \tag{4.50}
\]

**Proof.** We first derive the expression of the expected edge set size at time \(t+1\), conditioned on the random vector \(X\) as follows:

\[
E\left(\mathcal{E}_{\rho}^{(t+1)} \mid X\right) = \frac{1}{2} \sum_{i \in T_\rho(\kappa)} \psi_i - X_i \ E\left[\Delta_{i \in T(\kappa)}^{(t+1)} \mid X\right] + \frac{1}{2} \sum_{i \in T_\sigma(\kappa)} \psi_i \ E\left[\Delta_{i \notin T(\kappa)}^{(t+1)} \mid X\right] + \frac{1}{2} \sum_{i \notin T(\kappa)} \psi_i \ E\left[\Delta_{i \in T(\kappa)}^{(t+1)} \mid X\right] - \frac{1}{2} \sum_{i \in T_\rho(\kappa)} X_i \ E\left[\Delta_{i \notin T(\kappa)}^{(t+1)} \mid X\right]. \tag{4.51}
\]

Taking the expectation operator over the random vector \(X\), we get:

\[
E\left(\mathcal{E}_{\rho}^{(t+1)}\right) = \frac{1}{2} \sum_{i \in T(\kappa)} \psi_i \ E_X\left\{E\left[\Delta_{i \in T(\kappa)}^{(t+1)} \mid X\right]\right\} + \frac{1}{2} \sum_{i \notin T(\kappa)} \psi_i \ E_X\left\{E\left[\Delta_{i \notin T(\kappa)}^{(t+1)} \mid X\right]\right\} - \sum_{i \in T_\rho(\kappa)} E_X\left\{X_i \ E\left[\Delta_{i \in T(\kappa)}^{(t+1)} \mid X\right]\right\}. \tag{4.51}
\]

Substituting (4.37) and (4.38) in (4.51), and using the definition of \(\alpha_i\) and \(\beta_i\) in (4.48)
and (4.49), respectively, we get:

\[
\mathbb{E} \left[ |\mathcal{E}_p^{(t+1)}| \right] = \frac{1}{2} \sum_{i \in T(\kappa)} \psi_i \left\{ \mathbb{E} \left[ \Delta_i^{(t)} \right] + \alpha_i \right\} + \frac{1}{2} \sum_{i \not\in T(\kappa)} \psi_i \left\{ \mathbb{E} \left[ \Delta_i^{(t)} \right] + \beta_i \right\} \\
- \mathbb{E}_X \{ X_i \} \mathbb{E} \left[ \Delta_i^{(t)} \right] + \sum_{k=1 \atop k \neq i}^M \psi_k \varrho_k \mathbb{E}_X \{ X_i^2 \} \frac{1}{N(N-1)} \left( \mathbb{E}_X \{ X_i \} + \mathbb{E}_X \{ X_i^2 \} \right) \\
- \sum_{k \in T_\rho(\kappa) \atop k \neq i} \frac{\mathbb{E}_X \{ X_iX_k \}}{N} \left( \mathbb{E}_X \{ X_i \} (\varrho_k - \psi_k) + \mathbb{E}_X \{ X_k^2 \} \right) \frac{1}{N(N-1)} \right) \\
+ \sum_{k \in T_\rho(\kappa) \atop k \neq i} \psi_k \mathbb{E}_X \{ X_k \} \left( \varrho_k \mathbb{E}_X \{ X_i \} + \mathbb{E}_X \{ X_k^2 \} \right) \\
\frac{1}{N(N-1)} \right) (4.52)
\]

Using the definitions of \( \xi_k, \Phi_k(z), \Lambda_k(z) \) in (4.41), (4.39) and (4.40), respectively, we get:

\[
\mathbb{E} \left[ |\mathcal{E}_p^{(t+1)}| \right] = \mathbb{E} \left[ |\mathcal{E}_p^{(t)}| \right] + \frac{1}{2} \sum_{i \in T(\kappa)} \psi_i \alpha_i + \frac{1}{2} \sum_{i \not\in T(\kappa)} \psi_i \beta_i \\
- \frac{1}{2} \sum_{i \in T_\rho(\kappa)} \varrho_i \left( \mathbb{E} \left[ \Delta_i^{(t)} \right] + \sum_{k=1 \atop k \neq i}^M \xi_k - \sum_{k \in T_\rho(\kappa) \atop k \neq i} \Phi_k(1) + \sum_{k \in T_\rho(\kappa) \atop k \neq i} \Lambda_k(1) \right) . (4.53)
\]

Finally, using the definition of \( \gamma_i \) in (4.50), we get:

\[
\mathbb{E} \left[ |\mathcal{E}_p^{(t+1)}| \right] = \mathbb{E} \left[ |\mathcal{E}_p^{(t)}| \right] - \frac{1}{2} \sum_{i \in T_\rho(\kappa)} \varrho_i \left( \mathbb{E} \left[ \Delta_i^{(t)} \right] + \gamma_i \right) + \frac{1}{2} \sum_{i \in T(\kappa)} \psi_i \alpha_i + \frac{1}{2} \sum_{i \not\in T(\kappa)} \psi_i \beta_i . (4.54)
\]

Note that the expression of the expected edge set size evolution for the broadcast case is the same as the one in (4.47), but with \( \alpha_i, \beta_i \) and \( \gamma_i \) not having the terms representing
\( k \in \mathcal{T}_\sigma(\kappa) \).

After deriving these evolution formulae, we will study the receiver selection strategy maximizing the coding density in the next section.

### 4.6 Receiver Selection Strategy

From (4.47), we can infer that the value of the expected edge set size at time \( t + 1 \) is affected by two main components with respect to its value at time \( t \). The first component represents an expected reduction in the edge set size due to the possible elimination of the primary served vertices. This elimination of vertices results in the removal of both their adjacent edges at time \( t \), and the potential additions to these edges at time \( t + 1 \) if they were not eliminated. This component is reflected in the \( \mathbb{E} \left[ \Delta_i(t) \right] + \gamma_i \) term. The second component is the changes in the degrees of the remaining vertices, which are controlled by the coefficients \( \alpha_i \) and \( \beta_i \) for targeted and non-targeted receivers, respectively. We will thus investigate the effect of receiver selection on these two components.

#### 4.6.1 Vertex Elimination

Serving primary vertices in a transmission is expected to eliminate them from the primary graph after this transmission if their inducing receivers receive the transmitted packet. The elimination of these vertices results in the removal of their adjacent edges at time \( t \). It also results in the loss of the potential changes in their degrees due to the probable elimination of the other served vertices.

The loss of these vertices and their edges is a natural outcome of the recovery transmission process and is unavoidable. We cannot reduce the effect of this opportunity loss component by reducing the size of the primary targeted receiver set, because the vertex set will remain large, which could reduce the coding density. Also, since we are aiming to study the minimization of completion delay, it is not a good idea to maximize the
coding opportunities at the expense of reducing the number of served vertices, as this will tend to increase the completion delay. However, we can still reduce the effect of this loss component by serving the vertices with smaller degrees. The following theorem compares the expected vertex degrees of two receivers given the sizes of their Has and Wants sets.

**Theorem 4.8.** If $\psi_i - \psi_h = d_1$ and $\varrho_h - \varrho_i = d_2$, then:

$$E[\Delta_h] - E[\Delta_i] = \frac{d_1}{N} \left( 1 + \frac{\varrho_h \varrho_i}{N - 1} \right) + d_2 \left( \sum_{k=1}^{M} \frac{\psi_k \varrho_k}{N(N-1)} \right).$$  

(4.55)

**Proof.** From the expected vertex degree expression in (4.30), we have

$$E[\Delta_h] = \sum_{k=1}^{M} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k \varrho_h}{N - 1} \right) + \frac{\psi_i}{N} \left( 1 + \frac{\varrho_i \varrho_h}{N - 1} \right)$$

$$= \sum_{k=1}^{M} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k (\varrho_i + d_2)}{N - 1} \right) + \frac{\psi_h + d_1}{N} \left( 1 + \frac{\varrho_i \varrho_h}{N - 1} \right)$$

$$= \sum_{k=1}^{M} \frac{\psi_k}{N} \left( 1 + \frac{\varrho_k \varrho_i}{N - 1} \right) + \frac{\psi_h}{N} \left( 1 + \frac{\varrho_h \varrho_i}{N - 1} \right) + \frac{d_1}{N} \left( 1 + \frac{\varrho_i \varrho_h}{N - 1} \right)$$

$$+ d_2 \left( \sum_{k=1}^{M} \frac{\psi_k \varrho_k}{N(N-1)} \right)$$

$$= E[\Delta_i] + \frac{d_1}{N} \left( 1 + \frac{\varrho_h \varrho_i}{N - 1} \right) + d_2 \left( \sum_{k=1}^{M} \frac{\psi_k \varrho_k}{N(N-1)} \right).$$  

(4.56)

Theorem 4.8 proves that the vertices of the receivers with larger Wants sets and smaller Has sets have smaller expected degrees than those having smaller Wants sets and larger Has sets. The difference in the expected vertex degrees between two receivers is
linear in the difference between their Wants and Has set sizes. Consequently, targeting the receivers having larger Wants sets and smaller Has sets will reduce the loss of edges due to vertex elimination.

Note that the condition of having a large Wants set becomes equivalent to the condition of having a small Has set, on average, if the receivers have similar demand ratios. In the broadcast scenarios, these two conditions are always equivalent ($d_1 = d_2$). In these aforementioned cases, targeting the receivers with the largest Wants sets tends to minimize the edge loss effect due to vertex elimination.

### 4.6.2 Degree Evolution of Remaining Vertices

The following theorem describe an important relation between the two coefficients controlling the change of the remaining vertex degrees.

**Theorem 4.9.** The increase in the degrees of the remaining vertices of any receiver is larger when it is targeted than when it is not. In other words, $\alpha_i \geq \beta_i \ \forall \ i \in \mathcal{M}$.

**Proof.**

\[
\alpha_i = \sum_{k=1}^{M} \frac{q_i \psi_k q_k}{N(N-1)} + \sum_{k \in T_{\alpha}(k) \ k \neq i} \frac{q_k \psi_k (q_i + q_i)}{N(N-1)} - \sum_{k \in T_{\beta}(k) \ k \neq i} \frac{q_k}{N} \left(1 + \frac{(q_k - \psi_k + 1)(q_i + q_i)}{N-1}\right) 
\]

(4.57)

Re-arranging the above expression and using the definition of $\beta_i$ in (4.49), we get:

\[
\alpha_i = \beta_i + \sum_{k \in T(\kappa) \ k \neq i} \frac{q_i \psi_k q_k}{N(N-1)} + \sum_{k \in T_{\alpha}(k) \ k \neq i} \frac{q_k \psi_k (q_k + q_i)}{N(N-1)} + \sum_{k \in T_{\beta}(k) \ k \neq i} \frac{q_i \psi_k q_k - q_i q_k (q_k - \psi_k + 1)}{N(N-1)} 
\]

(4.58)

Since for $\psi_k > 0$, $q_k \geq q_k - \psi_k + 1$ and $\psi_k > q_k$, the last term in (4.58) is non-negative and the theorem follows.

Another important insight about the values of $\alpha_i$ and $\beta_i$ can be inferred from the analysis of their components $\Phi_k$ and $\Lambda_k$, as shown in (4.48) and (4.49). Since the terms
\[
\sum_{k \in \mathcal{T}_{i}(\kappa)} \Phi_k(q_i) \quad \text{and} \quad \sum_{k \in \mathcal{T}_{i}(\kappa)} \Phi_k(0)
\]
are subtractive terms from \(\alpha_i\) and \(\beta_i\), respectively, then selecting the receivers with smaller values of \(\Phi_k(q_i)\) and \(\Phi_k(0)\) to be primary targeted receivers increases the values of \(\alpha_i\) and \(\beta_i\), respectively. Now, if \(q_k < q_h, \psi_k > \psi_h\) and \(\varrho_k < \varrho_h\), we have:

\[
q_k (\varrho_k - \psi_k + 1) < q_h (\varrho_h - \psi_h + 1) \quad (4.59)
\]
\[
\Rightarrow \quad \Phi_k(q_i) < \Phi_h(q_i) \quad \text{and} \quad \Phi_k(0) < \Phi_h(0) \quad (4.60)
\]

Consequently, the receivers having larger erasure probabilities, larger Wants sets and smaller Has sets have smaller values of \(\Phi_k(q_i)\) and \(\Phi_k(0)\). In case of equal demand ratios at all receivers (including the broadcast scenarios), the three above conditions are equivalent on average. In other words, the receivers having smaller reception probabilities will on average have larger Wants sets and smaller Has sets. Consequently, maximizing the number of such receivers in the primary targeted receiver set maximizes the values of both \(\alpha_i\) and \(\beta_i\).

Since the terms \(\sum_{k \in \mathcal{T}_{i}(\kappa)} \Lambda_k(q_i)\) and \(\sum_{k \in \mathcal{T}_{i}(\kappa)} \Lambda_k(0)\) are additive terms to \(\alpha_i\) and \(\beta_i\), respectively, then selecting the receivers with larger values of \(\Lambda_k(q_i)\) and \(\Lambda_k(0)\) to be secondary targeted receivers increases the values of \(\alpha_i\) and \(\beta_i\), respectively. The values of \(\Lambda_k(q_i)\) and \(\Lambda_k(0)\) are larger for receivers having larger value of \(q_k\psi_k\). These are on average the receivers that fall in the intermediate ranges of \(q_k\) and \(\psi_k\). In general, we can assume that these receivers are the remaining receivers with largest Wants sets and erasure probabilities, after targeting the receivers having the largest Wants sets and erasure probabilities with primary packets. Although this assumption may not be always valid, we will show in Section 4.7 that it is sufficient to achieve an outstanding coding density evolution performance.

From the above theorem and discussion, we can infer that the remaining vertex degrees tend to be maximized by targeting the receivers, having the largest Wants sets and erasure
probabilities, with primary packets, then targeting the remaining receivers, having the largest Want sets and erasure probabilities, with secondary packets. We will refer to this strategy as the worst receiver layered targeting (WoRLT) strategy. This strategy both maximizes the coefficients $\alpha_i$ and $\beta_i$ for all $i \in \mathcal{M}$ and multiplies the maximum number of largest $\psi_i$ values with their larger coefficient $\alpha_i$.

4.6.3 Selection Strategy

From the above two theorems and discussion, we can draw the following observations:

**Observation 1:**
Targeting the receivers with largest Wants sets and smallest Has sets reduces the loss of edges due to vertex elimination. On average, this condition fits with the WoRLT strategy.

**Observation 2:**
The remaining vertex degrees tend to be maximized by applying the WoRLT strategy.

**Observation 3:**
Although targeting more receivers in general will cause more loss in the number of edges due to the probable vertex elimination from the graph, the reduction in the number of vertices in the graph will decrease the value of the denominator in the coding density expression with a much larger value $\left( \text{since } \mathbb{E}\left[\Delta_i^{(t)}\right] << |\mathcal{V}_\rho| - 1, \text{ except at the end of the recovery transmission phase} \right)$. This reduction along with the increase of the numerator due to Observation 2 are expected to cover this loss and thus the overall coding density of the system will be increased.

**Observation 4:**
From (4.47), we can see that the expected edge set size at $t + 1$ depends on its value at $t$. Consequently, maximizing the edge set size at time $t - 1$ to reach its value at time $t$ would also help in maximizing its value at time $t + 1$. In other words, following the strategy that maximizes the edge set size at each step helps its maximization in all future
steps until all the missing packet requests are satisfied.

From all the above observations, we conclude that the receiver selection strategy that is expected to maximize the coding density of the IDNC primary graph, in each step and throughout the duration of the frame transmission, is the WoRLT strategy. It is easy to infer that the same strategy increases the coding density in the broadcast IDNC graph. In this case, the worst receivers will be targeted with only primary packets since there are no secondary packets in the broadcast scenarios. We will test these findings through simulations in the next section.

4.7 Simulation Results

In this section, we test, through simulations, the performances of our identified strategies in improving the coding density during the transmission of a frame. We also compare them to other well-known strategies. The simulation scenario consists of transmitting 30 packets to 60 receivers having different packet reception probabilities ranging from 0.01 to 0.3, while maintaining the average packet reception probability \( p \) at 0.15. These reception probabilities are assumed to be fixed during the transmission of a frame but change from frame to frame during the simulation. The tested strategies in these simulations are:

- **RND**: Random clique selection.
- **MC**: Maximum clique selection.
- **MWC-R**: Maximum weighted clique in which the weight of vertex \( v_{ij} \) is defined as the reception probability of receiver \( i \) (i.e. \( q_i \)).
- **MWC-P**: Maximum weighted clique in which the weight of vertex \( v_{ij} \) is defined as a power of the demand set cardinality of packet \( j \) (i.e. \( \zeta_j^n \)).
• MWC-W: Maximum weighted clique in which the weight of vertex $v_{ij}$ is defined as a power of the Wants set cardinality divided by the reception probability (i.e. $\left(\frac{\psi_i}{\bar{q}_i}\right)^n$) of receiver $i$.

Although MWC-P is the only packet selection strategy simulated, we plot its performance in the same figure to compare it numerically with receiver selection strategies.

In MWC-W, the weight $\frac{\psi_i}{\bar{q}_i}$ gives more priority to the receivers with largest Want sets and erasure probabilities, as required in the WoRLT strategy. Moreover, the exponent $n$ represents the order of the bias given to the worst receivers. The same exponent is also used for the packet weighting in MWC-P to control the bias given to highly demanded packets. To perform layered targeting in MWC-W, the maximum weight clique search is first executed in the primary graph, then is executed again in the secondary subgraph adjacent to all the vertices of the primary chosen clique. To be fair to the other strategies in terms of measuring the coding density, we employ the same layered algorithm structure in all of them.

All figures represent the average coding density evolution in the primary graph along the transmissions of the recovery phase. We assume that the same strategy is always employed from the start of the recovery transmission phase until the delivery of all requested packets to their intended receivers. Figures 4.7(a) and 4.7(b) depict these evolutions for demand ratios of 0.5 and 1 (i.e. broadcast scenario), respectively. From both figures, we can draw the following observations. As expected, the proposed packet selection strategy considerably increases the coding density for the first 20-25% of the recovery transmissions due to the presence of packets with large Demands sets during this period. However, due to the reduction of these sets after several transmissions, the strategy cannot achieve the same improvement in terms of coding opportunities. Moreover, the restriction of serving such packets, when there are none, forces the algorithm to serve less vertices, which results in a lower coding density, as shown in the intermediate 22-30% of the recovery phase.
Figure 4.7: Average coding density evolution for $M = 60$ and $N = 30$
Towards the end of the phase, the number of vertices is naturally reduced and the Has sets becomes larger. From Theorem 4.8, we infer that the remaining vertices will have large expected degrees due to the large Has sets and small Wants sets of their inducing receivers. Consequently, the coding opportunities inside the graph will increase significantly. Moreover, the number of served vertices per transmission increase significantly, thus allowing the increase of the coding density. As shown in both figure, this near-completion effect occurs with all selection schemes.

We also observe in both figures that the coding density achieved by our proposed WoRLT receiver selection strategy considerably outperforms all other simulated receiver selection strategies. Moreover, we can see that the WoRLT strategy monotonically increases the coding density, which implies that it tends to maximize the coding density over the entire horizon of the recovery phase. Unlike the packet selection strategy, a single transmission can reduce the Wants set (and increase the Has set) of any receiver by at most one. Consequently, there will always exist some worst receivers during most of the recovery phase. Targeting these receivers will continuously increase the coding density in the IDNC primary graph.

Another important observation is that serving the maximum number of receivers at each transmission does not maximize the coding density, as shown in both figures. This can be explained from (4.56), proving the larger degrees of vertices belonging to receivers with smaller Wants sets. These large degree vertices tend to have very good adjacency among each other, thus creating larger cliques. Thus, serving the maximum number of vertices tends to be equivalent to targeting the receivers with smaller Wants sets. This strategy clearly conflicts with the evolution expression in Theorem 4.7, and thus it tends to minimize the edge set size rather than maximizing it. However, towards the end of the recovery phase, all techniques catch up with their efficiency similar to MWC-P.
4.8 Evolution of the Complete IDNC Graph

The complete IDNC graph is the graph representing all the coding opportunities among all lacked packets from all receivers whether they wanted them or not. If we assume that all the receivers want all the packets, the complete graph becomes a broadcast IDNC graph. Consequently, the evolution expressions for the complete IDNC graph in the multicast scenarios are equal to the evolution expressions for the broadcast IDNC graph, after replacing all $\mathcal{W}_i$ and $\psi_i$ in all these expressions by $\mathcal{L}_i$ and $\varphi_i$, $\forall i \in \mathcal{M}$.

Based on this result, all the identified packet and receiver selection policies, improving the coding opportunities and the coding density in the IDNC primary graph, directly applies to the complete IDNC graph. In other word, the packet selection strategy improving the coding opportunities in the complete IDNC graph is the one serving the packets that are lacked by the maximum number of receivers (i.e. with maximum $|\mathcal{U}_j|$). Moreover, the receiver selection strategy, continuously increasing the coding density of the complete IDNC graph for the entire recovery transmission phase, is the one serving the maximum number of receivers with largest Lacks sets and erasure probabilities.

4.9 Conclusion

In this chapter, we investigated the receiver and packet selection strategies that maximize the expected coding density in the IDNC graph. We first derived an expression for the exact evolution of the edge set size in the IDNC primary graph after the transmission of any arbitrary coded packet. From the expressions, we showed that serving packets requested by the largest number of receivers tends to maximize the coding density in the multicast and broadcast scenarios, especially in the beginning of the recovery transmission phase. Since the exact expression did not give insights on the receiver selection strategy, we derived an expected expression of the edge set size evolution after ignoring the Has, Lacks and Wants set contents and keeping their cardinalities. We employed this expected
expression to show that targeting the maximum number of receivers having the largest Wants sets and erasure probabilities tends to maximize both the expected number of coding opportunities and the expected coding density, for the multicast and broadcast scenarios. Simulation results showed that the identified packet selection strategy indeed improves the coding density in the beginning of the recover transmission phase. They also showed that the identified receiver selection strategy tends to maximize the coding density not only for one step, but for all future steps, thus achieving a continuous increase in the coding density. We finally showed that the identified packet and receiver selection strategies, maximizing the coding density in the IDNC primary graph, directly extend to the IDNC complete graph. We will use these results in optimizing the IDNC completion delay in the following chapters.
Chapter 5

IDNC Completion Delay

5.1 Motivation

As clarified in Chapter 1, one of the major weaknesses of IDNC is its vulnerability to result in a very high completion delay, if the scheduling of packet combinations is not optimized. In the broadcast scenario, IDNC can fall far behind the optimal FNC performance in terms of completion delay, which can reduce the value of IDNC and its benefits. In this chapter, we try to minimize this value degradation of IDNC by studying the problem of minimizing the completion delay in IDNC.

Unlike other delay parameters, the completion delay is a parameter that is not specific to each recovery transmission but rather a cumulative parameter throughout the entire recovery transmission phase. Consequently, the optimal network coding decisions, minimizing the completion delay, cannot be optimally determined locally per transmission but rather need to be jointly optimized along the entire recovery transmission phase. However, all preliminary thoughts and intuitions show that solving the completion delay minimization problem in IDNC over the entire recovery transmission phase is very difficult. As mentioned in Section 2.4, if we assume lossless channels during the recovery transmission phase, the problem of minimizing the completion delay in IDNC
becomes equivalent to the index coding problem, which has been proved to be NP-hard to solve [15, 16] and even NP-hard to approximate [38]. Now adding to that the fact of stochastic packet erasures over lossy channels, the problem becomes even worse.

According to this fact, we are only left with the option of designing an efficient heuristic, which solves the problem through local decisions (i.e. in each transmission), while observing the results of these decisions on future steps. One intuitive idea of this heuristic is to serve the maximum number of vertices in each transmission, which may both increase the coding opportunities in the IDNC primary graph and quickly deplete all its vertices. However, there have been no evidence or theoretical studies supporting this heuristic’s success in minimizing the IDNC completion delay.

In this chapter, we aim to study the IDNC completion delay in a more formal way, and to design an efficient heuristic according to the specifics of the problem. Consequently, we will first formulate the completion delay minimization problem as a stochastic shortest path problem, then employ this problem structure and the graph evolution findings in Chapter 4 to build our efficient heuristic.

### 5.2 Problem Formulation

The problem of minimizing the expected completion delay in IDNC can be formulated in the form of an SSP, explained in Chapter 2, as follows.

#### 5.2.1 State Space $S$

The states of the formulated SSP are defined by all possibilities of state feedback matrices that may occur during the recovery transmission phase. For state $s$, the matrix $F(s)$ represents the content of the Has, Lacks and Wants sets in $s$ (i.e. $\mathcal{H}_i(s)$, $\mathcal{L}_i(s)$ and $\mathcal{W}_i(s)$ $\forall i \in \mathcal{M}$) as defined by (3.1). According to its definition, the state space has a size of $|S| = O(2^{MN})$. 
5.2.2 Action Spaces $\mathcal{A}(s)$

In this formulation, the actions can be defined as all possible network coding decisions that could be made until an absorbing state of the SSP is reached. In some SSP (or MDP) formulations, all possible actions can be taken in all the states, and thus we can identify only one action space $\mathcal{A}$ for the whole SSP. However, in our case, the available network coding decisions depend on the state and some actions valid for one state are not valid for others. Consequently, we can define an action space $\mathcal{A}(s)$ for each state $s$ in the SSP.

Since the set of all network coding decisions in state $s$ are identified by all possible cliques in the IDNC graph constructed from $\mathbf{F}(s)$, we set the action space $\mathcal{A}(s)$ at state $s$ to be all cliques $\kappa(s) \in \mathcal{K}(s)$. It has been proved that the number of cliques in certain graphs grows exponentially with the number of vertices [13]. If we consider the set of maximal cliques only, this scaling still applies since the maximum number of maximal cliques in a graph with $n$ vertices to be $O\left(3^{n/3}\right)$. Thus, in both cases, the action set size of each state grows exponentially with $O(MN)$.

5.2.3 State-Action Transition Probabilities

To define the state-action transition probabilities $P_{\kappa(s)}(s, s')$ for an action $\kappa(s) \in \mathcal{A}(s)$, we first introduce the following two sets:

$$\mathcal{X} = \{i \in \mathcal{T}(\kappa(s)) \mid \varphi_i(s) > \varphi_i(s')\} \quad (5.1)$$
$$\mathcal{Y} = \{i \in \mathcal{T}(\kappa(s)) \mid \varphi_i(s) = \varphi_i(s')\} \quad (5.2)$$

The first set includes the targeted receivers whose Lacks sets have decreased from state $s$ to state $s'$, and thus have successfully received the IDNC packet generated from $\kappa(s)$. The second set includes the targeted receivers that have lost the IDNC packet generated from $\kappa(s)$ and thus their Lacks sets did not change. Based on the definitions of these
sets, $P_{\kappa(s)}(s, s')$ can be expressed as follows:

$$P_{\kappa(s)}(s, s') = \prod_{i \in \mathcal{X}} q_i \cdot \prod_{i \in \mathcal{Y}} p_i .$$  \hfill (5.3)

Figure 5.1 depicts the state representation and the action space for the example in Figure 3.1. It also depicts the possible transitions and their probabilities given that action $a_7$ is performed.

### 5.2.4 State-Action Costs

The expected completion delay is defined in SSP terms as the expected number of transitions in the process before arriving to an absorbing state. Since any transition (due to any action) takes one packet transmission, the cost paid by the process is one time-slot. Consequently, the costs of all actions in all states should be set to 1. In other words, $c(s, \kappa(s)) = 1 \ \forall \ \kappa(s) \in \mathcal{A}(s), s \in \mathcal{S}$. Consequently, minimizing the expected cost until absorption in this SSP will be equivalent to minimizing the number of transmissions until completion (i.e. completion delay).

Given this definition of costs, the value function $V_\pi(s)$ of any state $s$ given any policy
\( \pi \) becomes the expected number of transmissions starting from that state until all the receivers receive their requested packets, when policy \( \pi \) is followed.

### 5.2.5 SSP Solution Complexity

As explained in Section 2.5, the optimal policy of an SSP problem can be computed using the well-known policy iteration and value iteration algorithms. As explained in Section 2.5, the lowest complexity of these algorithms is \( \Theta(|S|^2|A|) \). According to the dimensions of \( S \) and \( A(s) \) described above, we conclude that computing the optimal policy is impossible in real-time for typical values of \( M \) and \( N \). Indeed, for a small network setting, with 10 receivers and 10 packets, the state space size is \( O(2^{100}) \) and the number of actions per state is \( O(3^{100/3}) \) (if only maximal cliques are considered). Even the simulation based technique proposed in [48] for finite MDPs, will not be able to compute the policy in real-time since its complexity still scales with \( |S| \).

### 5.3 Properties of Formulated SSP

#### 5.3.1 SSP Structure

Despite the complexity of solving the SSP problem formulated in Section 5.2, we will study its properties and structure to draw the characteristics of the policies that can efficiently minimize the expected completion delay. From Section 5.2, it is easy to infer that the SSP formulation has the following three properties.

**Property 5.1** (Uniform Cost).

*The costs of all actions in all states are all the same except for the absorbing states.*

**Property 5.2** (Non-singleton acyclicity).

*No state can be revisited once the process moves to one of its successor (also called child) state. Indeed, if some packets are received by some receivers when an action is taken at a*
given state, there is no means of going back with these receivers not having these packets. This makes the SSP formulation acyclic. However, a state can revisit itself (singleton cycles) if none of the targeted receivers by the taken action receives the IDNC packet.

**Property 5.3** (Non-increasing successor value functions).

*Since there are no cycles of size more than one, the successor states of a given state are all closer to the absorbing states than itself. Consequently, the expected cost to absorption starting from a given state is always greater than or equal to the expected costs to absorption starting from all its successor states.*

These three properties can be employed to draw the properties of the optimal policy \( \pi^* \) minimizing the expected completion delay at any given state \( s \) as follows. From the uniform cost in Property 5.1, we have:

\[
\pi^*(s) = \arg \min_{\kappa(s) \in \mathcal{A}(s)} \left\{ 1 + \sum_{s' \in S(s, \kappa(s))} P_{\kappa}(s, s') V_{\pi^*}(s') \right\}
\]

\[
= \arg \min_{\kappa(s) \in \mathcal{A}(s)} \left\{ \sum_{s' \in S(s, \kappa(s))} P_{\kappa}(s, s') V_{\pi^*}(s') \right\}
\]

\[
= \arg \min_{\kappa(s) \in \mathcal{A}(s)} \left\{ E_{\kappa(s)} [V_{\pi^*}(s')] \right\} \tag{5.4}
\]

where \( E_{\kappa(s)} \) is the expectation operator over the different transition probabilities when action \( \kappa(s) \) is taken. Thus, the optimal action at state \( s \) is the action minimizing the expectation of the optimal value functions of the successor states. From Properties 5.2 and 5.3, we know that all successors of state \( s \) are closer to the absorbing states (thus having less expected completion delays) except for itself. Consequently, the optimal action at state \( s \) is the one that has high probability of moving to the state(s) having the minimum expected residual completion delay (i.e. minimum mean time to absorption), given the optimal policy.

Now the problem is that there is no closed form expression for the optimal value
functions $V_{\pi^*}(s')$ in IDNC and thus there is no means of accurately computing it to determine the optimal policy without solving the SSP. However, based on the previous properties and facts, we can infer that the value of $V_{\pi^*}(s')$ for any $s'$, that is successor to state $s$, depends on two main factors:

- The closeness of the state’s Wants vector $\psi(s')$ to that of the absorbing states $\psi_0$. We say that a state $s'$ is closer to absorption than $s''$ if there is a theoretical possibility to reach an absorbing state from $s'$ in less number of transmissions compared to $s''$, regardless of their available actions.

- The availability of actions at state $s'$ and its successors that can bring the system faster to an absorbing state. This can translate into the number and size of the primary maximal cliques available as actions in state $s'$.

Indeed, if $\psi(s')$ is closer to $\psi_0$, we expect that the optimal value function for state $s'$ is smaller. However, this condition is not enough as we should also check the availability of efficient actions at this state that can bring the system faster to an absorbing state. In general, the successor states of $s$, whose primary graphs include more numerous and larger maximal cliques, have more chances of reaching the absorbing state faster than the others. Since all states $s'$ are successors of a same state $s$, their graphs are different variants of $G(s)$ depending on the vertices that have been served. Consequently, the action at state $s$, which can maximize the coding density in the IDNC primary graph at state $s'$ and for each of its vertices, will result in larger and more numerous primary maximal cliques. Based on these observation, we state that the policy that can efficiently reduce the expected completion delay in IDNC should aim, at any visited state, to both:

- Bring the system Wants vector the closest to the absorbing states vector $\psi_0$.

- Maximize the coding opportunities in the successor state’s primary graph.

If we can find a policy that can simultaneously achieve these two goals, we will employ it to design an algorithm to efficiently reduce the expected completion delay for IDNC.
investigate the existence of such policy, we need to study two important features of the problem, namely its geometric structure, and the evolution of coding density in the IDNC graph. The latter has been studied in details in Chapter 4. In the following section, we will focus on the former component.

5.3.2 Geometric Structure

In this section, we will explore the actions, which have high chances of moving the system Wants vector closest to that of the absorbing states. Given the representation of the SSP states by their Wants vectors, we can define a geometric structure as follows. Define an $M$-dimensional space $[\Psi_1, \ldots, \Psi_M]$, whose points $\Psi$ are identified by the coordinates of the vectors $\psi(s) \forall s \in S$. All states having the same Wants vector are located at the same point in the space. These states located at the same point differ from one another by their IDNC graphs. Also, all absorbing states are located at the all-zero point that we will denote by $\Psi_0$. Note that this geometric representation has the same non-singleton acyclicity property as the SSP (i.e. a point cannot be revisited after it is left).

Since at most one packet can be decoded by each receiver from any INDC transmission, the process can at most move along the dimensions of a hypercube defined by the points $\{\psi(s')|\psi(s') = \psi(s) - 1\}$, where $1$ is the all ones vector. In this case, the best action at any state is the one that can bring the process to the hypercube diagonal point. However, this action may not exist in the IDNC graph. Consequently, we need a method to estimate the closeness of other points to the absorbing point.

Figure 5.2 depicts the geometric structure of the example in Figure 5.1 after removing the fourth column (i.e. removing the fourth packet and the actions it appears in). Consequently, the process is at the point identified by the Wants vector $\psi(s) = [2, 1, 1]$. In this example, there are only five actions $a_1, a_2, a_3, a_5, a_6$, according to their notation in Figure 5.1. Assuming that the channels are lossless for one transmission, action $a_3$ will lead the process to point $[1, 1, 1]$ whereas action $a_5$ will lead it to point $[2, 0, 0]$. Although
\(a_5\) targets more receivers than \(a_3\), we can clearly see that \(a_3\) provide to the system with a non-zero probability of moving to the absorption point with one more erasure-free transmission, if there exist an IDNC packet targeting all three receivers at the state located at point \([1, 1, 1]\). One the other hand, with action \(a_5\), the probability of moving the the absorption point in one further step is strictly zero. This closeness to absorption in terms of Wants vectors is shown through the smaller geometric distance from point \([1, 1, 1]\) to \(\Psi_0\), compared to point \([2, 0, 0]\). We can infer from this example that minimizing the maximum entry in the Wants vector (i.e. \(\max_i\{\psi_i(s')\}\)) brings the process closer to the absorbing point. The philosophy behind this fact is that the receivers with largest Wants sets will impose their Wants set cardinalities as lower bounds on the completion delay. Consequently, serving these receivers first gives hope to reduce this lower bound at each step whereas ignoring them will not change the lower bound.

However, minimizing the maximum of the Wants vector entries is not enough to describe the actions with closest successor states to absorption. For example, actions \(a_1\) and \(a_3\) have the same value of \(\max_i\{\psi_i(s')\}\) but \(a_1\) brings the system closer to the absorbing point compared to \(a_3\) in terms of Wants vector, since it serves an additional
receiver with smaller Wants set. This is also reflected on the geometric distance from the two destination points to $\Psi_0$.

From the above example, we can conclude that, in order to bring the process closest to the absorbing point in terms of Wants vector, the sender should give more weight in serving the receivers with largest coordinate entries, while maximizing the number of served receivers with the smaller coordinate entries. This weighting can be done through a norm expression. For example, the $L_2$ norm (Euclidian distance) in the previous examples was representative of the state closeness to absorption in terms of Wants vectors. The larger the employed norm, the more biased the weighting is in giving service to the receivers with largest coordinate entries. Moreover, since we need to target as many receivers as possible given this weighting, we can restrict our action spaces to be the set of maximal cliques $K_m$ rather than the set of all cliques in the graph.

Now for erasure channels, the effect of packet erasures should be reflected on the geometric structure of the problem. Let $i$ and $k$ be two receivers having the same Wants set size but $p_i > p_k$. Consequently, $i$ will require on average more targeting attempts compared to $k$ in order to deplete its Wants set. Since we assume erasure probabilities do not change during the transmission of a frame, not targeting $i$ will add more delay to its individual completion and thus will have higher impact on the overall completion delay compared to $k$, especially when $\tilde{\psi}_i(s)$ is among the largest values in $\tilde{\psi}(s)$. According to these facts and the philosophy explained above for the erasure-free case, $i$ should be given higher priority of service than $k$ to reduce its higher impact on the completion delay. This can be done by representing $i$-th coordinate further from $\Psi_0$ than that of $k$ when using weighting through norms.

To do so, we define a channel weighted Wants vector $\tilde{\psi}(s) = [\tilde{\psi}_1(s), \ldots, \tilde{\psi}_M(s)]$, where $\tilde{\psi}_i(s) = \frac{\psi_i(s)}{q_i}$. Based on this new vector definition, we can redefine our space such that its points $\Psi$ are identified by the coordinates of the vectors $\tilde{\psi}(s)$ instead of $\psi(s) \forall s \in S$. In this case, the actions move the process within hyper-rectangles with
sides equal to $q_i^{-1}$ in the $i$-th dimension. The sender should then take the action that can reach the successor states with minimum $L_n$ norm. The following lemma shows that the optimal value function at any given state is upper and lower bounded by norms expressions of the state’s channel weighted Wants vector.

**Lemma 5.1.** The optimal value function at state $s$ is upper and lower bounded by the $L_1$ and $L_\infty$ norms of the channel weighted Wants vector at state $s$. In other words:

$$\|\tilde{\psi}(s)\|_\infty \leq V_{\pi^*}(s) \leq \|\tilde{\psi}(s)\|_1 .$$

(5.5)

**Proof.** It is known that the optimal value function for state $s$ (i.e. minimum completion delay starting from this state) is lower bounded by the expected completion delay of an imaginary rate optimal approach, which could be expressed as:

$$E\left[\max_{i \in \mathcal{M}} \{Z_i(s)\}\right] = E[\|Z(s)\|_\infty] ,$$

(5.6)

where $Z_i(s)$ is a random variable representing the number of transmission required by receiver $i$ for its individual completion when starting at state $s$, and $Z = [Z_1(s), \ldots, Z_M(s)]$. We also know that the optimal value function for state $s$ is upper bounded by the expected value of the sum of individual completion delays of each of the receivers, which could be expressed as:

$$E\left[\sum_{i \in \mathcal{M}} \{Z_i\}\right] = \sum_{i \in \mathcal{M}} E[Z_i] = \|E[Z]\|_1 ,$$

(5.7)

Consequently, we have:

$$\|E[Z(s)]\|_1 \geq V_{\pi^*}(s) \geq E[\|Z(s)\|_\infty] \geq \|E[Z(s)]\|_\infty .$$

(5.8)

The right most inequality results from Jensen’s inequality, given that the infinity norm
is a convex function. Since, the vector \( \tilde{\psi}(s) = \left[ \frac{\psi_1(s)}{q_1}, \ldots, \frac{\psi_M(s)}{q_M} \right] = E[Z(s)] \), the above inequality can be re-written as:

\[
\|\tilde{\psi}(s)\|_\infty \leq V_{\pi^*}(s) \leq \|\tilde{\psi}(s)\|_1.
\] (5.9)

This concludes the proof.

Since we know, from the definition of norms, that:

\[
\|\tilde{\psi}(s)\|_\infty \leq \|\tilde{\psi}(s)\|_n \leq \|\tilde{\psi}(s)\|_1 \quad \forall \ 1 < n < \infty,
\] (5.10)

we can infer, from Equations (5.9) and (5.10), that the \( L_n \) norms of the channel weighed Wants vectors are good representations of the state’s closeness to absorption.

From all the above structure and observations, we can draw a conclusion that an efficient policy, minimizing the expected completion delay, must always aim at each visited state \( s \) to reach a state \( s' \) that is located at the point with minimum distance to the absorbing point \( \Psi_0 \). Thus, this state should have the minimum \( \|\tilde{\psi}_i(s')\|_n \). Consequently, the receivers with larger values of \( \tilde{\psi}_i \) will have higher priority to be selected for transmission at state \( s \). Now, if we can show that this norm based selection of the receivers also maximizes the coding density in the successor state and its successors, then this norm based selection policy will efficiently reduce the IDNC completion delay, as explained in Section 5.3. To investigate this point, we will use our study of the receiver selection strategy maximizing the coding density evolution in the IDNC, as will be seen in the next section.

### 5.3.3 Evolution of Action Space

As stated in Section 5.3.1, one major factor that identifies the efficiency of an action at state \( s \) in minimizing the expected completion delay is its ability to foster efficient
actions at its expected successor state, which can bring the system faster to absorption. In general, the successor states of $s$, whose primary graphs include more numerous and larger maximal cliques, have more opportunities of reaching an absorbing state faster than the others. Since the graphs at these successor states of $s$ are different variants of $G(s)$, the action at state $s$, which can maximize the coding opportunities in the expected next visited state, will result in larger and more numerous primary maximal cliques. But since we aim to serve as many vertices as possible to reach completion faster, the efficient action should be maximizing the coding density in the IDNC primary graph.

In Chapter 4, we showed that the coding density of the IDNC primary graph is greatly improved and monotonically increased over the entire recovery transmission phase, when the worst receiver layered targeting (WoRLT) strategy is employed. For the primary graph, the WoRLT strategy can be executed through a norm minimization of the channel weighted Wants vector $\psi$. Indeed, such minimization will result in targeting the maximum number of receivers having the largest Wants sets and erasure probabilities. According to the discussion in Section 5.3.2, this policy perfectly matches the policy bringing the system the closest to the absorbing point $\Psi_0$.

To determine the set of secondary targeted receivers, we can apply a norm minimization of the channel weighted Wants vector $\tilde{\psi}$ within the IDNC secondary subgraph, adjacent to all selected vertices in the primary graph using norm minimization. Since the receivers with the largest Wants sets and erasure probabilities will be targeted with vertices from the primary graph, and since each receiver can have at most one vertex per clique, applying the norm minimization in the secondary subgraph, adjacent to the primary selected vertices, will result in targeting the remaining receivers with largest Wants sets and erasure probabilities. This approach conforms with the WoRLT strategy identified in Chapter 4. Moreover, it does not conflict with the policy bringing the process closest to absorption but rather fosters it in future steps. Indeed, serving these receivers with secondary packet will increase the coding opportunities of their remaining
primary (and secondary) vertices, which allows them to get served faster thus bringing the process closest to absorption.

Given the above facts, we conclude this section by stating that a norm minimization based receiver selection using the channel weighted Wants vector can efficiently reduce the average completion delay in IDNC and thus should be executed at each visited state until absorption. This norm minimization based selection approach is most efficient if it is first applied within the primary graph then applied on the secondary adjacent subgraph. We will thus design our proposed algorithms according to these guidelines in the following section.

5.4 Proposed Algorithms

According to the findings of Section 5.3, we propose a two-step maximal clique selection algorithm that should be executed at any visited state $s$. In the first step, the algorithm selects the maximal clique $\kappa^*_\rho(s)$ in the primary graph that targets receivers with larger channel weighted Wants set sizes, thus minimizing $\|\tilde{\psi}_i(s)\|_n$ for the expected successor state and maximizing the coding density of its primary graph. To further maximize the coding density in both the IDNC primary and complete graphs, the same process should be performed over the secondary subgraph, adjacent to all the vertices $\kappa^*_\rho$, to find $\kappa^*_\sigma(s)$. Each of these two steps can be implemented using a maximum weight clique selection algorithm.

5.4.1 Maximum Weight Clique Selection Algorithm

For each vertex $v_{ij}$ in the IDNC graph, we assign a weight $\left(\tilde{\psi}_i(s)\right)^n$, where $n$ is the order of the employed norm. We then execute a maximum weight clique selection algorithm on the primary graph. Since the algorithm will select vertices with largest $\left(\tilde{\psi}_i(s)\right)^n$ values, the selected maximum weight clique $\kappa^*_\rho$ will minimize the norm of the channel weighted
Wants vector of the expected successor state. After finding this clique, the secondary subgraph adjacent to all the vertices of $\kappa^*_\rho$ is extracted and the maximum weight clique selection algorithm is run on it to obtain $\kappa^*_\sigma$. When both cliques are found, the sender sends an IDNC packet that is generated by XORing all the source packets identified by the vertices in both cliques. After receiving the feedback from the receivers, the sender determines the successor state reached and the whole procedure is re-executed. This loop is run until all vertices in the primary graph are depleted.

It is well known that the maximum weight clique selection problem is NP-hard [21], and is hard to approximate [4]. On the other hand, there exist several polynomial time algorithms solving this problem for moderate size graphs ([80] and references therein). However, the complexity of these algorithms may still be prohibitive for several applications and network settings [80]. Consequently, we will design a simple heuristic in the next section to solve the problem with much lower complexity.

### 5.4.2 Maximum Weight Vertex Search Algorithm

In this section, we design a simple algorithm that performs clique selection, using a maximum weight vertex search. For this search to be efficient, the vertices’ weights must not only reflect the $\left(\tilde{\psi}_i(s)\right)^n$ values of their inducing receivers, but also their adjacency to vertices having large $\left(\tilde{\psi}_i(s)\right)^n$ values.

To design these vertices’ weights, we first define $a_{ij,kl}(s)$ as the adjacency indicator of vertices $v_{ij}$ and $v_{kl}$ in $G(s)$ such that:

$$a_{ij,kl}(s) = \begin{cases} 1 & \text{if } v_{ij} \text{ is adjacent to } v_{kl} \text{ in } G(s) \\ 0 & \text{otherwise} \end{cases} \quad (5.11)$$
We then define the weighted degree $\Delta_{ij}^w(s)$ of vertex $v_{ij}$ as:

$$\Delta_{ij}^w(s) = \sum_{\forall v_{kl} \in G(s)} a_{ij,kl}(s) \left( \bar{\psi}_k(s) \right)^n.$$  \hspace{1cm} (5.12)

Thus, a large weighted vertex degree reflects its adjacency to a large number of vertices belonging to receivers with large values of $\left( \bar{\psi}_k(s) \right)^n$. We finally define the vertex weight $w_{ij}(s)$ as:

$$w_{ij}(s) = \left( \bar{\psi}_i(s) \right)^n \Delta_{ij}^w(s).$$ \hspace{1cm} (5.13)

Consequently, a vertex $v_{ij}$ has a large weight when it both belongs to a receiver with large $\left( \bar{\psi}_i(s) \right)^n$ value and is adjacent to a large number of vertices having large $\left( \bar{\psi}_i(s) \right)^n$ values.

Let $G^r(s)$ be the subgraph in $G(s)$ only including all vertices connected to vertex $v$. We finally define $A(s)$ and $A_v(s)$ as the adjacency matrices of $G(s)$ and $G_v(s)$, respectively.

Based on these definitions, we can introduce our proposed heuristic algorithm, which operates only at the visited states. In each visited state $s$, the algorithm computes a primary and secondary maximal cliques $\kappa^*_\rho(s)$ and $\kappa^*_\sigma(s)$. At first, $\kappa^*_\rho(s)$ and $\kappa^*_\sigma(s)$ are empty sets. The algorithm starts by selecting the maximum weight vertex in $G_\rho(s)$ to be the source node in $\kappa^*_\rho(s)$. For each of the following iterations, the algorithm first recomputes the new vertex weights within the primary subgraph adjacent to all previously selected vertices in $\kappa^*_\rho(s)$, then adds the new maximum weight vertex to it. When there is no further primary vertices adjacent to all vertices in $\kappa^*_\rho(s)$, then the same process is repeated in the secondary subgraph adjacent to $\kappa^*_\rho(s)$, to build $\kappa^*_\sigma(s)$, until no vertices are remaining in the complete graph. The final maximal clique $\kappa^*(s)$ is thus the union of $\kappa^*_\rho(s)$ and $\kappa^*_\sigma(s)$. Once this clique is computed, the sender forms and sends an IDNC packet by XORing the source packets identified by the vertices in $\kappa^*(s)$. According to the received feedback, a new state is visited and the process is re-executed until an absorbing state is reached. The pseudo code for the algorithm is depicted in Algorithm 1.
Algorithm 1 Maximum Weight Vertex Search Algorithm

Require: $F(s)$ and $\tilde{\varphi}(s)$

Initialize $K^*(s) = \emptyset$.

Construct $\mathcal{G}(s)$ and $A(s)$.

while $\mathcal{G}_\rho(s) \neq \emptyset$ do
    Compute $\Delta_{ij}^{\rho}(s)$ and $w_{ij}(s)$ using (5.12) and (5.13).
    Select $v^* = \arg \max_{v_{ij} \in \mathcal{G}_\rho(s)} \{w_{ij}(s)\}$.
    Add $v^*$ to $\kappa^*_\rho(s)$.
    Set $\mathcal{G}(s) \leftarrow \mathcal{G}^{v^*}(s)$ and $A(s) \leftarrow A^{v^*}(s)$.
end while

while $\mathcal{G}(s) \neq \emptyset$ do
    Compute $\Delta_{ij}^{\sigma}(s)$ and $w_{ij}(s)$ using (5.12) and (5.13).
    Select $v^* = \arg \max_{v_{ij} \in \mathcal{G}(s)} \{w_{ij}(s)\}$.
    Add $v^*$ to $\kappa^*_\sigma(s)$.
    Set $\mathcal{G}(s) \leftarrow \mathcal{G}^{v^*}(s)$ and $A(s) \leftarrow A^{v^*}(s)$.
end while

$k^*(s) = k^*_\rho(s) \cup k^*_\sigma(s)$.

5.4.3 Complexity of the Heuristic Algorithm

According to the description of the maximum weight vertex search algorithm, the generation of one coded packet at the sender requires $O(M)$ iterations, since any maximal clique in the IDNC graph cannot be of size larger than $M$. Each of these iterations consists of $O(|V|)$ computations of the vertex weights, $O(\log |V|)$ computations for ordering the weights and $O(1)$ computation for XORing the resulting packet to the previously XORed packets. Since the vertex set size of the IDNC graph is $O(MN)$, the overall complexity of selecting a clique for a recovery transmission is $O(M(|V| + \log |V| + 1)) = O(M^2N)$. Since we know from Section 3.3.3 that the complexity of the construction and update of the graph itself is $O(M^2N)$, then the overall complexity of the IDNC heuristic algorithm, including both graph construction/update and clique selection is still $O(M^2N)$.

This complexity is much lower than that of the conventional maximum weight clique selection heuristics. However, it is considered high compared to the complexity of FNC, which is equal to $O(N)$. Nonetheless, this comparison is not completely fair as it equates the simple XOR operations in IDNC with the much more complicated operations over
large Galois field in FNC. Moreover, this complexity is experienced by the sender side, which may be tolerable due to the high processing power, cost, space and energy capabilities of the senders, such as the base stations of cellular or WiMAX networks. Remember from Chapter 3 that at the receiver side, the decoding complexity of FNC is $O(N^3)$, which is much higher than the $O(N^2)$ decoding complexity in IDNC. This increase in complexity at the receiver side is less tolerable due to the tight constraints and limitations on the size, processing power, and energy consumption of the receivers, in order to maintain their simplicity, low cost and battery lifetime.

### 5.5 Simulation Results

In this section, we present simulation results comparing the performance of our proposed algorithms to the following algorithms in both multicast and broadcast scenarios:

- **RND**: Random clique selection.

- **MC**: Maximum clique selection. This algorithm first selects the maximum clique $\kappa^\rho_{\text{max}}(s)$ in the primary graph then selects the secondary maximum clique from the secondary subgraph connected to $\kappa^\rho_{\text{max}}(s)$.

- **FNC**: Full network coding, which represents the global optimal completion delay in the broadcast scenarios.

For our proposed algorithms, we use different norms to test the effect of the selection bias on the algorithm’s performance. We consider the $L_1$, $L_2$, $L_3$, $L_5$ and $L_{10}$ norms. Also, for both our proposed algorithms and the MC algorithms, we test both the maximum weighted clique selection algorithm (denoted by “opt”) and the maximum weight vertex search algorithm (denoted by “srh”). In our simulations, we assume that different receivers have different packet erasure probabilities and different demand ratios, which
change from frame to frame while keeping the average erasure probability \((p)\) and average demand ratio \((\mu)\) constants.

Figures 5.3, 5.4, 5.5 and 5.6 depict the average completion delay performances of the maximum weight clique selection algorithm with different norms and compare them to those of the other algorithms, against the average demand ratio \(\mu\) (for \(M = 60, N = 30, p = 0.15\)), the number of receivers \(M\) (for \(\mu = 0.5\) and 1, \(N = 30, p = 0.15\)), the number of packets \(N\) (for \(\mu = 0.5\) and 1, \(M = 60, p = 0.15\)), and the average erasure probability \(p\) (for \(\mu = 0.5\) and 1, \(M = 60, N = 30\)), respectively.

From all these figures, we can draw the following conclusions:

- Our proposed maximum weight clique selection algorithm with all norms considerably outperforms the random and maximum clique selection algorithms in terms of average completion delay for all comparison parameters \((\mu, M, N\) and \(p)\).
Chapter 5. IDNC Completion Delay

![Graph](image)

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$

Figure 5.4: Average completion delay of different schemes and norms vs $M$
Figure 5.5: Average completion delay of different schemes and norms vs $N$
Figure 5.6: Average completion delay of different schemes and norms vs $p$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
• The $L_1$ norm algorithm degrades in performance compared to the other norms for all comparison parameters, due to its very conservative estimation of $V_{\pi^*}(s)$ values, as explained in Lemma 5.1.

• For norms higher than 1, the algorithms tend to converge to the same performance with the lowest completion delays achieved by the $L_3$ and $L_5$ algorithms. For greater norms like $L_{10}$, the performance slightly degrades for most comparison parameters, which means that we do not need higher bias levels.

• For the broadcast case ($\mu = 1$), the results show that our proposed algorithm almost achieves the optimal performance of FNC for all comparison parameters, with a maximum degradation of 1 transmissions. This results in a maximum degradation in the frame delivery duration (from the start of the frame transmission until its reception at all receivers) of 1.9%, only occurring at very large number of receivers. This near-optimal performance is achieved while fully preserving the benefits of INDC compared to FNC.

Figures 5.7, 5.8, 5.9 and 5.10 compare the average completion delay of our proposed optimal maximum weight clique selection (denoted by “opt”) to that of our proposed heuristic maximum weight vertex search (denoted by “srh”) algorithms for $L_3$, $L_5$ and $L_{10}$, as well as the maximum clique algorithm, using the same simulation parameters in Figures 5.3, 5.4, 5.5 and 5.6, respectively. For the maximum clique approach, the heuristic algorithm is the same as the one depicted in Algorithm 1 after replacing the $\tilde{\psi}_i$ value of each vertex by its absolute primary degree.

From all these figures, we can see that the heuristic algorithms perform very closely to the optimal clique selection algorithms for all norms and all comparison parameters, with a maximum degradation of less than 1.2 transmissions. This results in a maximum degradation in the frame delivery duration of 4.4%, compared to FNC, only occurring at very large number of receivers in the broadcast setting. We can also observe a considerable
improvement achieved by our proposed heuristic algorithm with all norms compared to both the optimal and heuristic maximum clique selection algorithms.

Finally, we can observe in Figures 5.7 and 5.8(a) that FNC still achieves a better completion delay performance than the heuristic clique selection algorithm in multicast scenarios with large numbers of receivers and large average demand ratios. This result also holds with the optimal clique selection algorithms for values of $M$ larger than the ones in Figure 5.8(a). For large number of receivers, this result occurs due to the increasing conflicts of coding opportunities between the different subsets of receivers. Indeed, if the number of receivers increases while keeping the number of packets constant, a larger number of receivers will be in conflict with whichever maximal clique chosen for transmission, thus making most coded packets non-instantly decodable for a larger subset of the receivers. In this case, FNC becomes more efficient in completing the transmissions
Figure 5.8: Average completion delay of opt and srh algorithms vs $M$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 5.9: Average completion delay of opt and srh algorithms vs $N$
Figure 5.10: Average completion delay of opt and srh algorithms vs \( p \)
faster as it always provides all receiver with new knowledge in each transmission. Even if some of this knowledge is not required, the continuous innovation provided to all the receivers in all the transmissions will still help them to complete the reception faster than IDNC.

As the demand ratio increases, the receivers tend to increasingly want more packets from the frame and thus the IDNC primary graph will become larger and will need more time to be depleted. On the other hand, the FNC performance does not change as it always delivers all the packets regardless of the demand ratio. Consequently, at one point, the number of transmissions required to deplete the increasing IDNC primary graph will exceed the number of transmissions required by FNC to deliver all the packets. After this switch point, this gap between FNC and IDNC gradually increases to reach its maximum in the broadcast scenario (when $\mu = 1$), at which FNC achieves the optimal completion delay performance.

5.6 Conclusion

In this chapter, we considered the problem of minimizing the completion delay of IDNC in wireless multicast and broadcast scenarios. We first formulated the minimum completion delay problem for IDNC as a stochastic shortest path (SSP) problem and showed the impossibility of solving the problem optimally using this formulation. Nonetheless, we were able to draw some theoretical guidelines of the IDNC algorithms that can efficiently reduce its completion delay, using the properties of the formulated SSP and the nature of the IDNC graph evolution. Based on these guidelines, we designed a maximum weight clique selection algorithm that efficiently reduces the IDNC completion delay in polynomial time. For further complexity reduction, we also designed a quadratic-complexity maximum weight vertex search heuristic, which efficiently tracks the optimal clique selection performance and outperforms the random and optimal maximum clique algorithms.
For broadcast scenarios, simulation results showed that our proposed algorithms almost achieve the globally optimal performance of full network coding, while preserving all the benefits and simplicity of IDNC.

The only remaining challenge against making our IDNC algorithm a very efficient yet very simple solution to implement in real networks is its need for graph update after each transmission using continuous and accurate feedback. This constraint may not be feasible in several network settings. In Chapters 7 and 8, we will study the problem of optimizing the average completion delay of IDNC in intermittent and lossy feedback environments.

Finally, we observed that, despite the considerably better performance of our proposed IDNC algorithms in most multicast settings, they tend to perform worse than FNC in the multicast scenarios having large demand ratios and/or large number of receivers. This worse performance becomes guaranteed in the broadcast scenarios, where FNC achieves the optimal performance. Although this loss in performance is shown to be minimal and is easily justifiable given the large gains offered by IDNC, some applications may require the absolute minimization of the completion delay, regardless of any other benefits. To achieve the maximum gain in such applications, we need to design an adaptive scheme selecting, between IDNC and FNC, the scheme achieving the smaller completion delay, according to the network and demand settings. We will focus on designing this adaptive scheme in the next chapter.
Chapter 6

Adaptive IDNC-FNC Selection

6.1 Motivation

In the previous chapter, we proposed efficient algorithms to reduce the average completion delay in IDNC, which outperform FNC in most multicast settings and achieve a near-optimal performance in the broadcast scenarios. Nonetheless, FNC still outperforms IDNC, not only in the broadcast scenarios but even in some multicast settings with large number of receivers and large average demand ratios. Although the loss in performance by our proposed IDNC algorithms is small compared to the achieved benefits, it may be mandatory in some applications to always finish the frame transmission the earliest regardless of any other considerations.

An example of such applications in cellular and WiMAX networks is the case of allocating a very small bandwidth to non-real time MBS applications, which may occur if the sender is overloaded or is servicing higher priority traffic. It is thus very important to utilize the available scarce bandwidth to its maximum in order to improve the system throughput and sever as much data flows as possible. In this case, minimizing the completion delay of a frame of MBS packets, regardless of the employed technique, becomes

---

Part of the work in this chapter is accepted for publication in IEEE/ACM Transactions on Networking, DOI:10.1109/TNET.2010.2091652
very important in maximizing the bandwidth efficiency, since the frame packets will be received in shorter time, thus allowing the transmission of new frames and new packets earlier.

In the above scenarios, an optimal network coding would have been a perfect solution to employ in the recovery phase. However, the optimal network coding, minimizing completion delay, is hard to find [16] and hard to approximate [38]. Thus, we are only left with IDNC and FNC as heuristic solutions to the main problem. Since each of the IDNC and FNC schemes usually outperforms the other in different receiver, demand and feedback settings, the continuous and rapid change of these settings in wireless networks may limit the achievable completion delay reduction, if only one scheme is always employed. This fact motivated us to design an adaptive selection scheme that adaptively selects, between IDNC and FNC, the scheme that is expected to achieve a smaller completion delay, given the network and demand settings.

In order to determine the better scheme for recovery transmissions in each frame, we should compute an a priori estimate of the completion delay for each scheme. Since it is very difficult to find exact completion delay expressions for both IDNC and FNC, we have to determine some performance metrics for both schemes, which can achieve a high chance of correct scheme selection, when compared to each other. In general, FNC has been widely studied and different metrics can be used to estimate its performance. However, IDNC does not have the same privilege. In the next section, we will inspect the different possible metrics that can be used for both IDNC and FNC.

### 6.2 FNC Completion Delay Metrics

One simple metric for the FNC completion delay is its number of lossless transmissions until completion. In FNC, all packets lacked by any receiver must be delivered to it and thus the number of lossless recovery transmissions needed by receiver $i$ to correctly
decode all lacked packets is equal to the cardinality of its Lacks set $|\mathcal{L}_i|$. Consequently, the lossless completion delay for FNC ($D_F$) is equal to:

$$D_F = \max_{i \in \mathcal{M}} |\mathcal{L}_i| = \max_{i \in \mathcal{M}} \varphi_i . \tag{6.1}$$

In case of lossy transmissions, the lossy FNC completion delay ($\tilde{D}_F$) is equal to the maximum of $M$ negative binomial random variables $NegBin(\varphi_i, q_i)$ [19]. It is difficult to derive an expression for the expected value of $\tilde{D}_F$ (i.e. $E\left[\tilde{D}_F\right]$). However, we know that:

$$E\left[\tilde{D}_F\right] = E\left[\max_{i \in \mathcal{M}} \{NegBin(\varphi_i, q_i)\}\right]$$

$$\geq \max_{i \in \mathcal{M}} \{E[\text{NegBin}(\varphi_i, q_i)]\}$$

$$= \max_{i \in \mathcal{M}} \left\{ \frac{\varphi_i}{q_i} \right\} . \tag{6.2}$$

The inequality in the second line results from Jensen’s inequality due to the convexity of the max function. We will use this lower bound of $E\left[\tilde{D}_F\right]$ as a metric for the lossy FNC performance.

### 6.3 IDNC Completion Delay Metrics

As explained above, an exact expression of the IDNC completion delay is not yet derived. In fact, deriving such expression seems to be extremely difficult due to both the complicated possibilities of targeting different subsets of receivers according to the cliques of the IDNC graph and the stochastic evolution of these possibilities. Consequently, we need to find simpler and derivable metrics that can estimate the performance of IDNC and make a proper scheme selection when compared to corresponding FNC metrics. This will be the target of this section.
6.3.1 Theoretical Foundation

Similar to FNC, one simple metric to consider would be the number of lossless recovery transmissions required to deplete the IDNC primary graph. This metric is similar to the number of index coding transmissions, achieved by the heuristic proposed in [11]. Define the complementary IDNC primary graph $G^c_\rho (V_\rho, E^c_\rho)$, such that $E^c_\rho = V_\rho \times V_\rho \setminus E_\rho$. In words, the complimentary IDNC primary graph is the graph having the same vertices of the IDNC primary graph $G_\rho$ and all possible edges that are not in $G_\rho$. In this case, the minimum lossless completion delay for IDNC (denoted by $D_I$) can be expressed as:

$$D_I = \chi (G^c_\rho). \quad (6.3)$$

where $\chi (G^c_\rho)$ is the chromatic number of the complimentary IDNC primary graph. In graph theory, the chromatic number of a graph is defined as the minimum number of colors that can be used to color all the vertices of this graph, such that no two adjacent vertices have the same color. Consequently, the chromatic number of a graph represents the minimum number of independent sets that include all the vertices of the graph. Since an independent sets in the complementary IDNC primary graph is equivalent to a clique in the original primary graph, $\chi (G^c_\rho)$ represents the minimum number of cliques covering all the vertices of the graph. In other words, it represents the minimum number of recovery transmissions or minimum completion delay of IDNC assuming lossless channels.

Since it is difficult to derive an expression for the IDNC completion delay $\bar{D}_I$ over lossy channels, we propose two methods to estimate its relative performance compared to FNC:

- **Method 1**: We can estimate the performance of IDNC through its minimum lossless completion delay $D_I = \chi (G^c_\rho)$ and compare it to the lossless FNC completion delay $D_F = \max_{i\in\mathcal{M}} \varphi_i$. Since finding the chromatic number of a graph is NP-hard, we need to find an approximation for $\chi (G^c_\rho)$. 
- **Method 2**: We can extend the IDNC primary graph representation, described in Chapter 3, by incorporating an average level of packet erasures inside it, thus generating a lossy IDNC primary graph model $\tilde{G}_\rho$. We then can estimate the chromatic number of the complimentary graph of $\tilde{G}_\rho$ and compare it to an approximation of the FNC lossy performance.

In both methods, we need to estimate the chromatic number of a graph. To do so, we propose the modeling of $G^e_\rho$ or $\tilde{G}^e_\rho$ as a random graph $G(\nu, \pi)$ having the same vertex set size $\nu$ or $\tilde{\nu}$ of $G^e_\rho$ or $\tilde{G}^e_\rho$, respectively, and a vertex adjacency probability $\pi$. If we can find this model, then we can apply the result in the following lemma, proved in [9], to approximate the chromatic number of $G^e_\rho$ or $\tilde{G}^e_\rho$.

**Lemma 6.1.** Almost every random graph $G(\nu, \pi)$, with $\nu$ vertices and a fixed probability $\pi$ ($0 < \pi < 1$) that any two vertices are adjacent, has a chromatic number that can be expressed as:

$$\chi(G(\nu, \pi)) = \left(\frac{1}{2} + o(1)\right) \log \left(\frac{1}{1 - \pi}\right) \frac{\nu}{\log \nu}. \quad (6.4)$$

Several approaches can be used to model the vertex adjacency of the IDNC primary graph by a uniform probability $\pi$ using the vertex adjacency conditions in either the lossless graph $G^e_\rho$ (Method 1) or a newly designed lossy graph $\tilde{G}^e_\rho$ (Method 2). Such approaches can differ in the amount of information used in computing $\pi$. To compare between the two above methods and determine the complexity and knowledge levels needed to obtain an efficient algorithm, we first propose Approaches I and II to derive $\pi_I$ and $\pi_{II}$ from $G^e_\rho$ according to Method 1. We then propose Approach III, in which we design $\tilde{G}^e_\rho$, according to Method 2, and compute its $\pi_{III}$ accordingly.

### 6.3.2 Approach I

In this approach, we propose to ignore both the vertices’ identities (their $ij$ indices) and the contents of the Has, Lacks and Wants sets of all receivers. This approach considers
only the graph vertex set size $\nu$, the system parameters $M$, $N$ and the cardinalities of the different sets. We consider this approach since the completion delay mainly depends on the cardinalities of the sets and not their contents as shown in Chapter 5.

Let $\mathcal{O}$ be the status descriptor of each MBS frame after the initial transmission phase, such that $\mathcal{O} = \{\varrho, \varphi, \psi, \zeta, \nu\}$. Given this status descriptor, we derive an expression for $\pi_i$ in the following theorem.

**Theorem 6.1.** Given $\mathcal{O}$, the probability $\pi_i$ of having any two vertices $v$ and $w$ adjacent in $G^c_\rho$ can be expressed as:

$$\pi_i = \frac{\psi \varphi^T}{N \nu} \left( 2 - \frac{\psi \varphi^T}{N \nu} \right) \left( 1 - \frac{\zeta (\zeta - 1)^T}{\nu (\nu - 1)} \right),$$

(6.5)

where $\mathbf{1}$ is the all ones row vector of appropriate dimensions.

**Proof.** Without loss of generality, we assume that $v$ is drawn from the graph’s vertex set before $w$. We also maintain the notation that $i, k$ are receiver indices and $j, l$ are packet indices. By inverting the adjacency conditions in the IDNC graph, described in Section 3.3.1, we know that two vertices $v_{ij}$ and $v_{kl}$ are adjacent in $G^c_\rho$ if and only if both following conditions hold:

- **C1:** $j \neq l \Rightarrow$ The two vertices do not represent the request of the same packet.
- **C2:** $j \notin \mathcal{H}_k$ OR $l \notin \mathcal{H}_i \Rightarrow$ At least one of the two vertices requests a packet that is not in the Has set of the other.

Since the events $\overline{C1}$ and $\overline{C2}$ are independent, we can express the vertex adjacency probability $\pi_i$ as:

$$\pi_i = P(\overline{C1}|\mathcal{O}) \ P(\overline{C2}|\mathcal{O}) = \left( 1 - P(C1|\mathcal{O}) \right) P(C2|\mathcal{O}) \ .$$

(6.6)

where $C1$ is the opposite condition of $\overline{C1}$, described in Section 3.3.1. For any two vertices
Chapter 6. Adaptive IDNC-FNC Selection

122

and \(w\), since we ignore the vertices’ identities, we get:

\[
\mathbb{P}(C_1 | \mathcal{D}) = \sum_{j=1}^{N} \mathbb{P}(v \to j | \mathcal{D}) \mathbb{P}(w \to j | \mathcal{D})
\]

\[
= \sum_{j=1}^{N} \frac{\zeta_j}{\nu} \frac{\zeta_j - 1}{\nu - 1} = \frac{\zeta(\zeta - 1)^T}{\nu(\nu - 1)},
\quad (6.7)
\]

where “\(v \to u\)” means “vertex \(v\) is induced by entity \(u\)” (\(u\) being a receiver or a packet).

Define event \(A\) as the event representing the request of \(v\) for a packet that is in the Lacks set of \(w\). Also define event \(B\) as the vice versa of event \(A\). Since we ignore the vertices' identities and the contents of the different sets, we can derive \(\mathbb{P}(A | \mathcal{D})\) as follows:

\[
\mathbb{P}(A | \mathcal{D}) = \sum_{k=1}^{M} \mathbb{P}(A | \mathcal{D}, w \to k) \mathbb{P}(w \to k | \mathcal{D}),
\quad (6.8)
\]

where

\[
\mathbb{P}(w \to k | \mathcal{D}) = \sum_{i=1}^{M} \mathbb{P}(w \to k | \mathcal{D}, v \to i) \mathbb{P}(v \to i | \mathcal{D})
\]

\[
= \sum_{i \neq k}^{M} \frac{\psi_k}{\nu - 1} \frac{\psi_i}{\nu} + \frac{\psi_k - 1}{\nu - 1} \frac{\psi_k}{\nu}
\]

\[
= \frac{\psi_k}{\nu(\nu - 1)} \left( \sum_{i=1}^{M} \psi_i - 1 \right) = \frac{\psi_k}{\nu(\nu - 1)} (\nu - 1) = \frac{\psi_k}{\nu},
\quad (6.9)
\]

Substituting (6.9) in (6.8), we get:

\[
\mathbb{P}(A | \mathcal{D}) = \sum_{k=1}^{M} \frac{\varphi_k}{N} \frac{\psi_k}{\nu} = \frac{\psi \varphi^T}{N\nu},
\quad (6.10)
\]

Similarly, we can derive an expression for \(\mathbb{P}(B | \mathcal{D})\) as follows:

\[
\mathbb{P}(B | \mathcal{D}) = \sum_{i=1}^{M} \mathbb{P}(B | \mathcal{D}, v \to i) \mathbb{P}(v \to i | \mathcal{D}) = \sum_{i=1}^{M} \frac{\varphi_i}{N} \frac{\psi_i}{\nu} = \frac{\psi \varphi^T}{N\nu},
\quad (6.11)
\]
Since packet erasures at different receivers are independent, the two events A and B are independent of each other. Now, from the definition of $C_2$, we get:

$$P(C_2|D) = P(A \cup B|D) = \frac{\psi \varphi^T}{N\nu} \left(2 - \frac{\psi \varphi^T}{N\nu}\right).$$

(6.12)

The theorem follows from substituting (6.7) and (6.12) in (6.6).

### 6.3.3 Approach II

One drawback of the previous approach is the need to compute $\pi_i$ for each frame since it depends on the cardinalities of the feedback sets. In this section, we consider a simpler approach that ignores these cardinalities in addition to the vertices’ identities. Consequently, this approach considers only the vertex set size $\nu$, the system parameters $(M, N)$ and the parameters of the packet request-erasure random process $(\mu_i, p_i)$. For simplicity of the analysis in this section, we will assume that all the receivers have an equal average packet erasure probability $p$, and a demand ratio equal to the average demand ratio $\mu$. Although, this simplification in the model should result in a lower performance, we will show in Section 6.5 that this degradation is negligible.

Given that we ignore all the feedback matrix information, including the set cardinalities, and only consider $\nu$, $\zeta$ becomes a random vector where $\zeta_N = \nu - \sum_{v=1}^{N-1} \zeta_v$. Any realization $\zeta' = \{\zeta'_1, \ldots, \zeta'_N\}$ of $\zeta$ is just a random subset of $\nu$ vertices from a pool of $MN$ candidate vertices. This random subset results from the packet request-erasure random process in the initial transmission phase. Consequently, we can introduce the following lemma.

**Lemma 6.2.** Given $\nu$, $\zeta$ is a multivariate hypergeometric distributed random vector. In other words,

$$P(\zeta = \zeta' | \nu) = \prod_{u=1}^{N-1} \left(\frac{M}{\zeta'_u}\right) \cdot \left(\nu - \sum_{v=1}^{N-1} \zeta'_v\right) \cdot \left(\frac{MN}{\nu}\right)^{-1},$$

(6.13)
where $\zeta_u' \in \{0, 1, \ldots, M\}$ for all $u \in \{1, \ldots, N-1\}$ and

$$
\binom{n}{k} = \begin{cases} 
\frac{n!}{k!(n-k)!} & 0 \leq k \leq n \\
0 & \text{otherwise}
\end{cases}
$$

(6.14)

**Proof.** During the initial transmission phase, packet $j$ will induce one vertex in $G^c_\rho$ if a receiver has requested and lost this packet. This event might occur for each of the $M$ receivers with probability $\mu p$. Since $\zeta_j$ is equal to the number of vertices in $G^c_\rho$ induced by $j$, $\zeta_j$ is the sum of $M$ Bernoulli trials and thus is a binomial random variable $Bin(M, \mu p)$. This applies for all $j \in \{1, \ldots, N\}$. In general, these random variables are independent. However, given that the total number of vertices in $G^c_\rho$ is equal to $\nu = \sum_{v=1}^N \zeta_v$, their sum is restricted. This restriction can be expressed as the dependence of one of these variables on the all the $N - 1$ other ones. Without loss of generality, we can assume that this variable is $\zeta_N$ which is set to $\nu - \sum_{v=1}^{N-1} \zeta_v$. The other $N - 1$ random variables become thus independent and can take any values. If the sum of these taken values exceeds $\nu$, $\zeta_N$ will be negative, and thus the probability of this event should be set to zero.

Based on these facts, and given the definition of the binomial coefficient in (6.14), the probability $P\left(\zeta = \zeta' \mid \sum_{v=1}^N \zeta_v = \nu\right)$ can be expressed as follows:

$$
P\left(\zeta = \zeta' \mid \sum_{v=1}^N \zeta_v = \nu\right) = \frac{P\left(\zeta = \zeta', \sum_{v=1}^N \zeta_v = \nu\right)}{P\left(\sum_{v=1}^N \zeta_v = \nu\right)}
= \frac{P\left(\zeta_1 = \zeta_1', \ldots, \zeta_{N-1} = \zeta_{N-1}', \zeta_N = \nu - \sum_{v=1}^{N-1} \zeta_v'ight)}{P\left(\sum_{v=1}^N \zeta_v = \nu\right)}
= \prod_{u=1}^{N-1} P\left(\zeta_u = \zeta_u'\right) P\left(\zeta_N = \nu - \sum_{v=1}^{N-1} \zeta_v'\right)
= \prod_{u=1}^{N-1} P\left(\zeta_u = \zeta_u'\right) \frac{P\left(\zeta_N = \nu - \sum_{v=1}^{N-1} \zeta_v'\right)}{P\left(\sum_{v=1}^N \zeta_v = \nu\right)}.
$$

(6.15)
Substituting with the binomial distributions expressions, we get

\[
P\left(\zeta = \zeta' \mid \sum_{v=1}^{N} \zeta_v = \nu\right) = \prod_{u=1}^{N-1} \left(\begin{array}{c} M \\ \zeta'_u \end{array}\right) (\mu p)^{\zeta'_u} (1 - \mu p)^{M - \zeta'_u} \\
\times \left(\nu - \sum_{v=1}^{N-1} \zeta'_v\right) (\mu p)^{\nu - \sum_{v=1}^{N-1} \zeta'_v} (1 - \mu p)^{M - \nu + \sum_{v=1}^{N-1} \zeta'_v} \\
\times \left(MN\right)^{\nu} (\mu p)^{\nu} (1 - \mu p)^{MN - \nu}^{-1}
\]

\[
= \prod_{u=1}^{N-1} \left(\begin{array}{c} M \\ \zeta'_u \end{array}\right) \cdot \left(\nu - \sum_{v=1}^{N-1} \zeta'_v\right) \cdot \left(MN\right)^{-1}.
\]

This expression is the probability mass function of a multivariate hypergeometric distribution, which concludes the proof.

Based on the above lemma, we can introduce the following theorem, which sets the adjacency probability in the complimentary IDNC primary graph \( G^c_{\rho} \), given the knowledge of only \( \nu, M, N, p \) and \( \mu \).

**Theorem 6.2.** Given \( \nu \), the probability \( \pi_{II} \) of having two vertices adjacent in \( G^c_{\rho} \) can be expressed as:

\[
\pi_{II} = p (2 - p) \frac{M(N - 1)}{MN - 1}.
\]

**Proof.** Since the two adjacency conditions in \( G^c_{\rho} \) (\( \overline{C1} \) and \( \overline{C2} \)) are independent, we can express the vertex adjacency probability \( \pi_{II} \), given \( \nu \) as:

\[
\pi_{II} = P\left(\overline{C1}|\nu\right) P\left(\overline{C2}|\nu\right)
\]

\[
= (1 - P\left(C1|\nu\right)) \cdot (1 - P\left(C2|\nu\right))
\]

where \( C1 \) and \( C2 \) are the opposite conditions of \( \overline{C1} \) and \( \overline{C2} \), illustrated in Section 3.3.1. \( C1 \) expresses the request of the two vertices for the same packet. Since we ignored the
vertices’ identities, we have:

\[ P(C1|\nu, \zeta' = \zeta') = \sum_{m=1}^{N} \frac{\zeta'_m (\zeta'_m - 1)}{\nu (\nu - 1)}. \]  

(6.22)

where \( \zeta_N = \nu - \sum_{\nu=1}^{N-1} \zeta'_\nu \). Thus, we can express \( P(C1|\nu) \) as follows:

\[ P(C1|\nu) = E_{\zeta|\nu} \left( \sum_{m=1}^{N} \frac{\zeta'_m (\zeta'_m - 1)}{\nu (\nu - 1)} \right) \]

\[ = \sum_{m=1}^{N} E_{\zeta|\nu} \left( \frac{\zeta'_m (\zeta'_m - 1)}{\nu (\nu - 1)} \right). \]  

(6.23)

From Lemma 6.2, we have \( \forall \ m \in \{1, \ldots, N - 1\} \):

\[ E_{\zeta|\nu} \left( \frac{\zeta'_m (\zeta'_m - 1)}{\nu (\nu - 1)} \right) = \sum_{\zeta'_m=0}^{M} \frac{\zeta'_m (\zeta'_m - 1)}{\nu (\nu - 1)} \]  

\[ \times \frac{\prod_{u=1}^{N-1} \left( \begin{array}{c} M \\ \zeta'_u \\ \end{array} \right) \left( \nu - \sum_{u=1}^{N-1} \zeta'_u \right)}{\left( \begin{array}{c} M \\ \nu \end{array} \right)} \]

\[ = \frac{M(M - 1)}{MN(MN - 1)} \times \sum_{\zeta'_m=0}^{M} \sum_{z_{m-2}=0}^{M-2} \frac{\prod_{u=1}^{N-1} \left( \begin{array}{c} M \\ \zeta'_u \\ \end{array} \right) \left( \zeta'_u - 2 \right) \left( \nu - \sum_{u=1}^{N-1} \zeta'_u \right)}{\left( \begin{array}{c} M \\ \nu - 2 \end{array} \right)} \]

\[ = \frac{M - 1}{N(MN - 1)}. \]  

(6.24)

The last line in (6.24) is obtained since the preceding line is a summation of a new multivariate hypergeometric probability mass function over all its sample space. For \( \zeta_N \), the same result can be derived using the same approach. Substituting in (6.23), we get:

\[ P(C1|\nu) = \frac{M - 1}{MN - 1}. \]  

(6.25)

C2 means that both vertices have correctly received the packet requested by each other. Since the reception of a packet from a receiver is independent of the reception of another packet by another receiver, and since we ignored the vertices’ identities and the
details of the feedback matrix, we get:

\[ P(C2|\nu) = 1 - (1 - p)^2 = p(2 - p). \]  \hfill (6.26)

The theorem follows from substituting (6.25) and (6.26) in (6.21).

We can see that the obtained expression of \( \pi_{\text{II}} \), using this approach, depends only on \( M, N \) and \( p \). These parameters generally vary with much smaller rate compared to the frame rate. This results in a lower computational rate of \( \pi_{\text{II}} \) compared to that needed for \( \pi_{\text{I}} \) in Approach I.

Note that we assumed equal packet erasure probabilities and demand ratios to be able to exploit the result in [9]. However, these assumptions are not true in practice. In Section 6.4, we will explain how to incorporate the practical erasure and demand values when employing this approach.

### 6.3.4 Approach III

In the previous two approaches, we ignored the packet erasures that might occur at different receivers when estimating the IDNC performances. In this section, we aim to consider these erasure possibilities in the estimation model to test whether this achieves a better performance than the previous two approaches. Since we do not know, at the selection time, the erasure realization that will occur during the recovery transmission phase, we will assume that an average number of erasure events will occur at each of the receivers. Thus, we need to find metrics for the expected completion delay \( E[\bar{D}_I] \) in IDNC.

Our approach to find such a metric is to both incorporate an average number of packet erasure events per receiver inside the IDNC graph and maintain the lossless channel assumption. We know that receiver \( i \) requires on average \( a_i = \frac{q_i}{i} \) transmissions to get one packet. Consequently, receiver \( i \) is expected to detect all its requested and lost
Chapter 6. Adaptive IDNC-FNC Selection

packets after $a_i\psi_i = \tilde{\psi}_i$ transmissions targeting it. Consequently, we will design our lossy IDNC graph model $\tilde{G}_\rho$ such that it has on average $\tilde{\psi}_i = a_i\psi_i$ vertices induced by $i$. These vertices should not be adjacent in the graph so that they represent different transmissions.

Based on the above description, we can build graph $\tilde{G}_\rho$ by replacing each vertex $v_{ij}$ in $G_\rho$ with $\Psi_{ij}$ vertices having the same identity $ij$, where:

$$P(\Psi_{ij} = \lfloor a_i \rfloor) = a_i - \lfloor a_i \rfloor$$

$$P(\Psi_{ij} = \lceil a_i \rceil) = a_i - \lfloor a_i \rfloor$$

where $\lfloor a_i \rfloor$ and $\lceil a_i \rceil$ are the floor and ceiling values of $a_i$, respectively. It is clear that $E[\Psi_{ij}] = a_i$. Consequently, the expected number of vertices induced by receiver $i$ in $\tilde{G}_\rho$ is equal to $a_i\psi_i$. The adjacency conditions in $\tilde{G}_\rho$ are similar to that of $G_\rho$ except that the vertices of the same identity must not be adjacent. Thus, we modify the adjacency conditions such that two vertices $v_{ij}$ and $v_{kl}$ are adjacent in $\tilde{G}_\rho$ if only one of the two following conditions holds:

- $\tilde{C}1$: $j = l$ AND $i \neq k$.

- $\tilde{C}2$: $j \in H_k$ AND $l \in H_i$.

Having the graph constructed, we can compute its complimentary graph $\tilde{G}_\rho^c$, randomize it as we did in the two previous approaches and employ the chromatic number of the corresponding random graph as a metric for the lossy IDNC performance.

To randomize $\tilde{G}_\rho^c$, we will ignore the vertices’ identities and the contents of the Has and Lacks sets only. In other words, the contents of the Wants sets will be considered in our computations. Since $\tilde{\nu}$ is defined as the number of vertices in $\tilde{G}_\rho^c$, the status descriptor
\[ \bar{\mathcal{D}} = \left\{ \bar{\Psi} = \begin{bmatrix} \bar{\Psi}_{ij} \end{bmatrix}, \theta, \bar{\nu} \right\}, \text{ where:} \]

\[ \bar{\Psi}_{ij} = \begin{cases} \lfloor a_i \rfloor \text{ or } \lceil a_i \rceil & \forall i, j \in \mathcal{W}_i \\ 0 & \forall i, j \notin \mathcal{W}_i \end{cases} \]  

(6.29)

Given this status descriptor, we present an expression for \( \pi_{III} \) in the following theorem.

**Theorem 6.3.** Given \( \bar{\mathcal{D}} \), the probability \( \pi_{III} \) of having any two vertices \( v \) and \( w \) adjacent in \( \bar{G}_c^\rho \) can be expressed as:

\[ \pi_{III} = 1 - \left( \frac{\text{Tr} \left[ \left( \bar{\Psi}^T \Theta \right) \bar{\Psi} \right]}{\bar{\nu}(\bar{\nu} - 1)} + \left( \frac{\left( \bar{\Psi}^T \theta \right) \bar{\nu}^T}{N\bar{\nu}} \right)^2 \right), \]  

(6.30)

where \( \text{Tr}[B] \) is the trace of matrix \( B \) and \( \Theta = [\Theta_{ij}] \) is defined as:

\[ \Theta_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}. \]  

(6.31)

**Proof.** Without loss of generality, we assume that \( v \) is drawn from the graph’s vertex set before \( w \). In this proof, we will maintain the notation that \( i, k \) are receiver indices and \( j \) is a packet index. The events \( \bar{\mathcal{C}}_1 \) and \( \bar{\mathcal{C}}_2 \) are mutually exclusive since Condition \( \bar{\mathcal{C}}_1 \) implies that \( j \in \mathcal{W}_k \) AND \( l \in \mathcal{W}_i \), which totally contradicts Condition \( \bar{\mathcal{C}}_2 \). Consequently, we can express \( \pi_{III} \) as:

\[ \pi_{III} = 1 - P \left( \bar{\mathcal{C}}_1 \cup \bar{\mathcal{C}}_2 \left| \bar{\mathcal{D}} \right. \right) \]

\[ = 1 - \left( P \left( \bar{\mathcal{C}}_1 \left| \bar{\mathcal{D}} \right. \right) + P \left( \bar{\mathcal{C}}_2 \left| \bar{\mathcal{D}} \right. \right) \right) \]  

(6.32)
Since we ignore the vertex identities, we can derive $P(C_1|\tilde{D})$ as follows:

$$P(C_1|\tilde{D}) = \sum_{i=1}^{M} \sum_{j=1}^{N} P(v \rightarrow (i,j), w \rightarrow (M \setminus (i,j))|\tilde{D})$$

$$= \sum_{i=1}^{M} \sum_{j=1}^{N} P(w \rightarrow (M \setminus (i,j)|\tilde{D}), v \rightarrow (i,j)) P(v \rightarrow (i,j)|\tilde{D})$$

$$= \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k \neq i}^{M} \frac{\tilde{\Psi}_{kj}}{\nu - 1} \tilde{\Psi}_{ij} = \frac{\text{Tr} \left[ \left( \tilde{\Psi}^T \Theta \right) \tilde{\Psi} \right]}{\nu(\nu - 1)} \quad (6.33)$$

Define event A as the event representing the request of $v$ for a packet that is in the Has set of $w$. Also define event B as the vice versa of event A. Since we ignore vertices identities, we can easily show, using similar derivations as in (6.8) and (6.11), that:

$$P(A|\tilde{D}) = P(B|\tilde{D}) = \left( \frac{1\tilde{\Psi}^T}{\nu} \right)^T$$

$$P(C_2|\tilde{D}) = P(A \cap B|\tilde{D}) = \left( \frac{1\tilde{\Psi}^T}{\nu} \right)^2 \quad (6.35)$$

The theorem follows from substituting (6.33) and (6.35) in (6.32).

Note that this approach involves more information and requires more complexity to construct the lossy IDNC graph model $\tilde{G}_\rho$.

Since we have derived $\pi_I$, $\pi_{II}$ and $\pi_{III}$ for Approaches I, II and III, respectively, we can now compute three metrics $\chi(\mathcal{G}(\nu, \pi_I))$, $\chi(\mathcal{G}(\nu, \pi_{II}))$ and $\chi(\mathcal{G}(\tilde{\nu}, \pi_{III}))$ for the IDNC completion delay performance, by substituting the derived expressions of $\pi_I$, $\pi_{II}$ and $\pi_{III}$, respectively, in (6.4), and using $\tilde{\nu}$ for the last metric. Having these metrics derived, we are now ready to introduce our adaptive IDNC-FNC selection scheme that we will refer
to as the *adaptive network coding (ANC)* scheme.

## 6.4 Adaptive Network Coding (ANC) Scheme

In the light of the results obtained in the previous sections, we will describe the proposed ANC scheme and discuss some of its implementation issues.

As explained earlier, ANC must select, between IDNC and FNC, the scheme that is expected to achieve a smaller completion delay in each frame, according to the system, demand and feedback parameters for this frame. Note that for broadcast scenarios, FNC is already known to be optimal [47]. So our focus will be only on multicast scenarios. For each frame, the ANC scheme selects either IDNC or FNC by comparing corresponding performance metrics of the two schemes. The scheme having the smaller value for the chosen metrics is selected to be executed for this frame.

The proposed ANC algorithm is illustrated in Algorithm 2. If the current frame is a broadcast frame, then the FNC scheme is directly employed for its recovery transmission phase, since it will always outperform the IDNC scheme. If the current frame is a multicast frame, then the algorithm computes and compares the following pairs of metrics depending on the employed approach.

- For Approach I, $\chi(G(\nu, \pi_I))$ is compared to $\max_{i \in M} \varphi_i$.
- For Approach II, $\chi(G(\nu, \pi_{II}))$ is compared to $\max_{i \in M} \varphi_i$.
- For Approach III, $\chi(G(\tilde{\nu}, \pi_{III}))$ is compared to $\max_{i \in M} \{a_i \varphi_i\}$.

The ANC then selects the scheme, whose metric is smaller, to be employed for the entire recovery transmission phase of this frame.

Note that for Approach II in Section 6.3.3, we assumed equal packet erasure probabilities and demand ratios to be able to exploit the result in [9]. However, these assumptions
Algorithm 2 The ANC Algorithm

if $\mu_i = 1 \forall i \in \mathcal{M}$ then
Run FNC.
else if Approach I is employed then
$\nu \leftarrow \sum_{i=1}^{M} \psi_i$.
$\pi_i$ is computed from (6.5).
$\chi(G(\nu, \pi_i))$ is computed from (6.4).
if $\chi(G(\nu, \pi_i)) \leq \max_{i \in \mathcal{M}} \varphi_i$ then
RUN IDNC.
else
RUN FNC.
end if
else if Approach II is employed then
$\nu \leftarrow \sum_{i=1}^{M} \psi_i$.
$p \leftarrow \frac{1}{M} \sum_{i=1}^{M} p_i$.
$\pi_{II}$ is computed from (6.20).
$\chi(G(\nu, \pi_{II}))$ is computed from (6.4).
if $\chi(G(\nu, \pi_{II})) \leq \max_{i \in \mathcal{M}} \varphi_i$ then
RUN IDNC.
else
RUN FNC.
end if
else if Approach III is employed then
$\tilde{\Psi}_{ij} \leftarrow \lceil a_i \rceil \text{ or } \lfloor a_i \rfloor \forall i, j \in \mathcal{W}_i$.
$\tilde{\nu} \leftarrow \sum_{i=1}^{M} \sum_{j \in \mathcal{W}_i} \tilde{\Psi}_{ij}$.
$\pi_{III}$ is computed from (6.30).
$\chi(G(\tilde{\nu}, \pi_{III}))$ is computed from (6.4).
if $\chi(G(\tilde{\nu}, \pi_{III})) \leq \max_{i \in \mathcal{M}} \{a_i \varphi_i\}$ then
RUN IDNC.
else
RUN FNC.
end if
end if
are not true in practice. Consequently, if Approach II is employed in our proposed algorithm, we will set $p$ in (6.20) to be to the average of the receivers’ packet erasure probabilities. We will also employ the actual average demand ratio $\mu$ of all receivers. Although this will affect the accuracy of the metric $\chi(G(\nu, \pi_{II}))$, we will show in the following section that the algorithm performance is still satisfactory.

Finally, the value of the $o(1)$ term employed in (6.4) for each of the three approaches will be empirically determined in Section 6.5.1 in order to achieve the best ANC selection performance.

## 6.5 Simulation Results

In this section, we test the performance of our proposed ANC algorithm through simulations. The simulation scenario is quite similar to the multicast scenario used in the previous chapters but with some increase in either the number of receivers or demand ratio, to get some ranges where FNC outperforms IDNC. Similar to the previous chapters, the packet erasure probability $p_i$ of each receiver changes per frame during the simulation time, taking values from 0.01 to 0.3, while keeping the average erasure probability constant at 0.15. Also, the demand ratio $\mu_i$ of each receiver changes with time while maintaining the average demand ratio $\mu$ constant. The results obtained in the following figures are computed over 2000 frames for each reading.

### 6.5.1 Study of the $o(1)$ Term

In the chromatic number expression (6.4), the term $o(1)$ tends to zero as the number of vertices $\nu$ tends to infinity. Since our IDNC graphs have finite numbers of vertices, we run a study on the $o(1)$ term, in order to determine the values that achieve the best selection performance for each of the three approaches. The metric employed to evaluate the different values of this term is the selection success probability defined as the number of
trials in which the ANC algorithm succeeds in selecting the scheme with smaller number of recovery transmissions, divided by the total number of trials (which is equal to 2000 trials).

Figures 6.1, 6.2 and 6.3 depict the selection success probabilities for Approaches I, II and III, respectively. In each figure, the upper subfigure depicts the selection success probability against $M$ for $N = 30$ and $\mu = 0.6$, whereas the lower subfigure depicts the same metric against $\mu$ for $M = 80$ and $N = 30$. For each subfigure, we test different values of $o(1)$ ranging from zero to one.

For Approach I, we can see from Figure 6.1(a) that the highest selection success probabilities are obtained at $o(1) = 0.3$ for all values of $M$. From Figure 6.1(b), we can see that $o(1) = 0.2$ dominates the others for all values of $\mu$. We then run a more focused simulation in the range $[0.2,0.3]$ and found that the best performance for all values of $M$ and $\mu$ is achieved at $o(1) = 0.24$, which makes it the best choice to employ for Approach I.

For Approach II, we can see from Figure 6.2(a) that $o(1) = 0.7$ and $o(1) = 0.8$ alternate in dominance over smaller and larger values of $M$, respectively. From Figure 6.2(b), we can see that $o(1) = 0.7$ dominates the performance for all values of $\mu$, and $o(1) = 0.8$ achieves a very close performance to $o(1) = 0.7$. We then run a more focused simulation in the range $[0.7,0.8]$ and found that the best performance for all values of $M$ and $\mu$ is achieved at $o(1) = 0.73$, which makes it the best choice to employ for Approach II.

Finally, $o(1) = 0.4$ dominates the performance for different values of $M$, as depicted in Figure 6.3(a). Figure 6.3(b) shows that the best selection performance is obtained at $o(1) = 0.3$ and $o(1) = 0.4$, for all values of $\mu$. We then run a more focused simulation in the range $[0.3,0.4]$ and found that the best performance for all values of $M$ and $\mu$ is achieved at $o(1) = 0.39$, which makes it the best choice to employ for Approach III.

We finally note that the selection success probabilities drops to around 70% in the middle range of $M$ and $\mu$. As will be shown in the next section, this drop occurs when the
Figure 6.1: Empirical study of the $o(1)$ term for Approach I
Figure 6.2: Empirical study of the $o(1)$ term for Approach II
Figure 6.3: Empirical study of the $o(1)$ term for Approach I
IDNC and FNC schemes achieve close performances. Consequently, the overall average performance of the ANC scheme is not significantly affected by this drop.

6.5.2 Performance Testing

In this section, we compare the performance of our three proposed ANC approaches to both IDNC and FNC for different numbers of receivers and demand ratios. As a comparison reference, we define the fictitious optimal selection scheme (denoted by OPT in the figure legends) as the one that always selects the correct recovery scheme achieving the smaller number of transmissions. We obtain the performance of this fictitious algorithm by simulating the IDNC and FNC algorithms and selecting at the end of each frame the smaller completion delay.

For Approaches I, II and III, Figures 6.4, 6.5 and 6.6 depict, respectively, the average completion delay achieved by the IDNC, FNC, optimal selection and ANC schemes. The upper subfigures 6.4(a), 6.5(a) and 6.6(a) illustrate the performance comparisons against $M$ for $N = 30$ and $\mu = 0.6$. The lower subfigures 6.4(b), 6.5(b) and 6.6(b) illustrate the performance comparisons against $\mu$ for $M = 80$ and $N = 30$.

We can observe from all the figures that all our proposed ANC approaches achieve an average performance that is always lesser or equal to the average performance of the better scheme. Moreover, this performance is achieved by our proposed selection metrics, which rely on random graph approximations, without adding any extra completion delay jitter to that achieved by the original selected scheme, even at the points where the performance of both schemes are very close and thus a degradation was expected.

Furthermore, we can observe that the performance of our proposed ANC approaches do not degrade much at the points with lower selection success probabilities shown in Figures 6.1, 6.2 and 6.3. This result is explained by the fact that the performance of both schemes are very close at these points, which greatly reduces the effect of wrong selection on the overall average performance. We can also see that our proposed ANC approaches
Figure 6.4: Performance of Approach I vs $M$ and $\mu$ for $o(1) = 0.24$
Figure 6.5: Performance of Approach II vs $M$ and $µ$ for $o(1) = 0.73$
Figure 6.6: Performance of Approach III vs $M$ and $\mu$ for $o(1) = 0.39$
almost achieve the optimal selection performance with maximum degradation between 0.5 to 1 transmission.

Finally, we conclude from all the figures that considering erasures in the selection process of the ANC scheme does not add much value as the lossless approximations already achieve the same or slightly better performance. Moreover, the results show that we do not need to compute $\pi$ every frame as was suggested in Approach I, because Approach II almost achieves the same performance with lower computation rate of $\pi$.

6.6 Conclusion

In this chapter, we designed an adaptive network coding scheme that combines the completion delay reduction gains of both IDNC and FNC, thus achieving the performance of the better scheme in all the ranges of number of receivers and demand ratios. The proposed scheme selects, between IDNC and FNC, the scheme that is expected to achieve the smaller completion delay. The core of this adaptive selection scheme lies in our derived performance metrics for the IDNC completion delay, which achieve efficient selection when compared to appropriate FNC metrics. To compare between different complexity levels and information requirements, we presented three approaches to derive the IDNC metrics. In the first two approaches, we first modeled the lossless IDNC primary graph by two random graphs with different parameters, using different levels of knowledge about the original graph. We then derived two IDNC completion delay metrics from the parameters of these two random graphs using a well-known expression of the random graph chromatic number.

To test the effect of packet erasures on the adaptive selection process, we proposed a third approach, in which we first designed a new lossy IDNC primary graph representation, by incorporating an average level of packet erasures inside the graph. We then derived a lossy IDNC performance metric using the same methodology employed for the
two lossless metrics. For the three considered approaches, simulation results showed that our proposed scheme almost achieves the completion delay performance that could be obtained by the optimal selection between IDNC and FNC. They also showed that this result can be achieved without adding extra performance fluctuation, without considering packet erasures and with a low parameter computation rate.
Chapter 7

IDNC with Intermittent Feedback

7.1 Motivation

So far, we have studied the problem of minimizing the IDNC completion delay, assuming that the IDNC graph is continuously and accurately updated after each recovery transmission, through instantaneous and lossless feedback. This assumption may be valid in some applications, networks and access technologies. Examples of these cases can be networks with out-of-band dedicated signalling channels and time division duplex (TDD) based networks with packets large enough to fill the whole MBS section in the downlink frame. In the latter example, the receivers will be able to respond to each packet by ACK packets in the uplink frame, either separately or piggy-backed to their uplink packets.

Nonetheless, several other network settings and applications may not be capable of providing such continuous feedback after every transmission. They rather refrain from sending any acknowledgements for a subsequent set of transmissions then send one feedback packet, acknowledging all the packets in these transmissions. We will refer to such feedback as intermittent feedback. One clear example of intermittent feedback is the same TDD based cellular or WiMAX setting explained above, in which more than one packet can be transmitted in one downlink frame. In this case, the sender will receive
an accumulation of all packet acknowledgments for every downlink frame in its subsequent uplink frame. Other examples of intermittent feedback scenarios are window-based flow controlled applications and the transmission of commands to sensor or robots with limited feedback capabilities.

Intermittent feedback may not be caused only because of the network or application setting. In the above TDD example, the packets can be large enough to fill the MBS section of the frame, but still the system would not exhaust each uplink frame with acknowledgements from all receivers. It would rather prefer to send one feedback packet every group of frames, acknowledging all received packets. This reduction in the feedback frequency is an important feature in window-based flow controlled applications to reduce network overloading and congestion events. It is also important to reduce the energy consumption in sensors and robots.

If IDNC is to be employed in any of the above examples, the sender must make blind IDNC decisions during the feedback silence time. These blind decisions, made without accurate knowledge of the IDNC graph status, will definitely affect the IDNC completion delay. In this chapter, we will study the problem of minimizing the IDNC completion delay in intermittent feedback scenarios. Since the example of TDD based cellular and WiMAX systems is the one of most interest to us in this thesis, we will start by formulating this problem as an SSP, similar to Chapter 5, assuming constant and synchronous intervals of feedback intermittence that we will refer to as the feedback periods ($T_f$). In the TDD example, a feedback period corresponds to the downlink frame duration. At the end of the chapter, we will discuss the validity of our designed solutions to operate in other network settings characterized by non-periodic and/or asynchronous intermittent feedback.

In intermittent feedback scenarios, we can distinguish two types of completion delays, namely, the reception completion delay and the completion feedback delay. The former delay is defined as the exact number of transmissions until all receivers receive all their
requested packets. This is the same as the completion delay definition, illustrated in Chapter 2, and thus we will call it completion delay for short. The latter delay is defined as the number of recovery transmissions until the sender gets completion feedbacks from all the receivers. This completion feedback delay is an integer multiple of the feedback period $T_f$. The formulation presented in the next section actually minimizes the expected feedback delay. Nonetheless, we will show from the properties of the formulation that it inherently minimizes the reception completion, which will be considered in our proposed solution design.

7.2 Problem Formulation

The minimum expected completion feedback delay problem for IDNC with intermittent feedback can be formulated as an SSP problem as follows.

7.2.1 State Space $S$

Similar to the original IDNC problem discussed in Chapter 5, the states of the intermittent feedback problem can be defined as all possibilities of SFM $F(s)$ that may occur during the recovery transmission phase. However, these states are different from the original IDNC problem in that the transition from one state to the following occurs after $T_f$ transmissions, when the sender obtains the feedback from the receivers at the end of the feedback period.

Despite of this difference, the cardinality of the state space is still $|S| = O(2^{MN})$ since all situation can still be visited despite the larger number of transmissions between state transitions.
7.2.2 Action Spaces $\mathcal{A}(s)$

In this problem, actions are taken at the beginning of each feedback period, after collecting feedback from all the receivers. In such intermittent feedback environment, an action consists of $T_f$ cliques from $\mathcal{G}(s)$, which cover the $T_f$ transmissions until the next feedback collection. Consequently, for each state $s$, the action space $\mathcal{A}(s)$ consists of all $T_f$ clique combinations in the graph $\mathcal{G}(s)$. Note that the cliques in one action may have vertex overlaps because serving a vertex does not guarantee the reception of its packet by its inducing receiver, and thus can be re-attempted. Moreover, we cannot restrict ourselves to the set of maximal cliques since the late transmissions in the feedback period may consist of sub-cliques of a maximal clique, due to the earlier serving of some of its vertices in the previous transmissions within the same feedback period. According to this description, the cardinality of state $s$ action space $|\mathcal{A}(s)|$ is equal to $\binom{|\mathcal{K}(s)|}{T_f}$.

7.2.3 State-Action Transition Probabilities

To define the state-action transition probability $P_a(s, s')$ for an action $a \in \mathcal{A}(s)$, we define $\Gamma_i(s, a)$ as the number of cliques, from the $T_f$ cliques chosen for action $a$, that include a vertex induced by $i$. Since each clique can include at most one vertex induced by the same receiver, $\Gamma_i(s, a) \leq T_f$ and represents the number of transmissions targeting receiver $i$ in action $a$. Also, let $\Upsilon_i(s, s') = \varphi_i(s) - \varphi_i(s')$, which represents the number of primary and secondary packets that should be received by receiver $i$, if the system transitions from state $s$ to $s'$. It is obvious that for any valid transition, $\Upsilon_i(s, s') \leq \Gamma_i(s, a)$.

Given these definitions, we can express $P_a(s, s')$ as follows:

$$
P_a(s, s') = \begin{cases} 
\prod_{i=1}^{M} q_i^{\Upsilon_i(s, s')} p_i^{(\Gamma_i(s, a) - \Upsilon_i(s, s'))} & 0 \leq \Upsilon_i(s, s') \leq \Gamma_i(s, a) \leq T_f \forall i \in \mathcal{M} \\
0 & \text{otherwise}.
\end{cases}
$$

The above expression ensures that the state-action transition probability is set to zero,
thus nullifying the possibility of $s'$ being a direct successor state of $s$, if:

- Any receiver lacks more packets in $s'$ than $s$ (when $\Upsilon_i(s, s') < 0$).
- Any receiver receives more packets than the number of times it has been targeted (when $\Upsilon_i(s, s') > \Gamma_i(s, a)$).
- Any receivers receives more than $T_f$ packets or is targeted more than $T_f$ times (when $\Upsilon_i(s, s')$ of $\Gamma_i(s, a)$ is greater than $T_f$).

### 7.2.4 State-Action Costs

Any action taken in the process represents $T_f$ packet transmissions and thus the completion feedback delay is increased by $T_f$ for any taken action. Consequently, in order to minimize the mean completion feedback delay, the cost of all actions in all states should be set to $T_f$. In other words, $c(s, a) = T_f \forall a \in \mathcal{A}(s), s \in \mathcal{S}$.

### 7.2.5 SSP Solution Complexity

According to the SSP formulation described above, the size of the state space $\mathcal{S}$ is equal to that of the original continuous feedback problem, studied in Chapter 5. One the other hand, the action space $\mathcal{A}(s)$ of any state $s$ is $\frac{(|K|-1)!}{T_f!(|K|-T_f)!}$ times larger than the continuous feedback one. Consequently, finding the optimal solution of the intermittent feedback problem is more complicated than the continuous feedback one.

### 7.3 Properties of the SSP Formulation

According to the above formulation, we can infer that the SSP has the same three properties of the original continuous feedback problem as follows:

**Property 7.1 (Uniform Cost).**

*The costs of all the actions in all the states are the same except for the absorbing states.*
Property 7.2 (Non-singleton acyclicity).

No state can be revisited once the process moves to its successor states. This makes the process acyclic except from singleton cycles representing the revisiting of the same state if none of the targeted receivers by any of the $T_f$ transmissions receives its targeting packet(s). Note that these singleton cycles become less probable, since each action includes several transmissions not only one as in the continuous feedback case. Consequently, it is very unlikely that none of the receiver get any packet after these several transmissions.

Property 7.3 (Non-increasing successor value functions).

Since the successor states of any state are all closer to the absorbing states than itself, the expected cost to absorption at this parent state is always greater than or equal to the expected costs to absorption at all its successor states.

Since the properties and structure of the intermittent feedback problem are the same as the continuous feedback one, we can define a similar geometric space for it. In this space, the process can transition along the $i$-th dimension of a hyper-rectangle with a distance $d \in \{ q_i^{-1}, 2q_i^{-1}, \ldots, \Gamma_i(s,a)q_i^{-1} \leq T_f q_i^{-1} \}$ per action, according to the number of received instantly decodable packets at $i$. These transitions inside this hyper-rectangle can be looked at as a resultant of $T_f$ separate transitions, each of which depending on the set of targeted receivers in its corresponding transmission and whether they received it. Since single transmissions occur at different times, they cannot be combined to achieve synergetic results. Consequently, maximizing the traveled distance after $T_f$ transmissions can be achieved by maximizing the transition of each single transmission. According to the study in Section 5.3, this can be done by employing the WoRLT strategy in each transmission. On one hand, the traveled distance per transmission will be maximized as in the continuous feedback case and on the other hand the coding density will be also maximized to provide longer transitions in the remaining steps. This will indeed result in maximizing the overall traveled distance by the end of the feedback period. Since the coding density is monotonically increasing when the WoRLT strategy is always employed,
the resulting coding density at the end of the feedback period will be also maximized.

The only difficulty to directly implement this approach in the intermittent feedback problem is that the sender should perform these sequential clique selections for $T_f$ transmissions without knowing the outcome of the attempted and un-acknowledged transmissions within the feedback period. To be capable of applying the above strategy in each of the transmissions of the intermittent feedback problem, the sender needs to make blind decisions about the changes in the IDNC graph’s vertices and edges after each transmission without feedback. In the following sections, we introduce four different approaches to deal with these vertices and introduce the modified algorithms accordingly.

### 7.4 Vertex and Graph Update Approaches

In this section, we present four vertex elimination and graph update approaches that could be applied to the IDNC graph after each un-acknowledged transmission within the feedback period. These approaches will help turning the problem of jointly selecting $T_f$ cliques into sequential clique selection problems after vertex and/or graph updates, thus allowing the use of the algorithms proposed in Section 5.4.

#### 7.4.1 Full Vertex Elimination (FVE)

In this approach, all attempted vertices in each transmission within the feedback period are eliminated from the graph. Consequently, at each visited state $s$, the sender will select $T_f$ fully disjoint cliques from $G(s)$ for the $T_f$ transmissions. Each of these cliques is chosen using the algorithms in Section 5.4 after the elimination of all the vertices of the previous cliques. At the feedback instants, the sender adjusts its graph according to the received feedback and then re-executes the above process in the following feedback period. In case there are no remaining vertices in the graph while the system is still within the feedback period, the sender retransmits the previously generated cliques, after the
previous feedback instant, in the remaining transmissions until the arrival of the new feedback. Note that if the erasure probability of all receivers are less than 0.5, this approach can be viewed as a maximum likelihood estimation of the intermediate states within the feedback period.

The advantage of this approach is that it guarantees full innovation for the targeted receivers in each transmission due to the use of disjoint cliques. Retransmissions of the previously served vertices occur only when the process is already near completion and have actually served most of the vertices.

**7.4.2 Full Vertex Elimination with Graph Update (FVE-GU)**

This approach is a variant of the previous approach, in which not only all attempted vertices are eliminated but also the sender updates the vertex adjacency in the IDNC graph assuming all attempted vertices are correctly received at their intended receivers. With this assumption, the new coding opportunities, arising from the reception of previous packets, are reflected in the graph. This approach may result in selecting better and larger cliques in the subsequent transmissions, which may target more receivers and help the process reach completion faster. However, for the vertices that were not actually received, this approach risks of generating non-instantly decodable packets for their inducing receivers. We will consider this approach to study the effect of the above trade-off.

**7.4.3 Stochastic Vertex Elimination (SVE)**

In this approach, the attempted vertices are eliminated from the graph probabilistically, according to the reception probabilities of their inducing receivers. When a vertex is attempted in a transmission within the feedback period, the sender keeps this vertex in the graph with probability $p_i$ and eliminates it with probability $q_i$. Similar to FVE, the sender adjusts its graph at each feedback instant and re-execute the above process in
the following feedback period. In case there are no remaining vertices in the graph while
the system is still within the feedback period, the sender retransmits the previously
generated cliques, after the previous feedback instant, in the remaining transmissions
until the arrival of the new feedback.

This approach better represents the reception probabilities of packets compared to
FVE and thus can re-attempt the non-received vertices in an earlier stage compared to
FVE. However, it falls in the risk of re-attempting received vertices in case the sender
decided to keep them in the graph. Again, this trade-off may or may not result in a
better completion delay.

7.4.4 Stochastic Vertex Elimination with Graph Update (SVE-
GU)

This approach is a variant of SVE in which the sender updates the vertex adjacency in the
IDNC graph assuming the correct reception of the stochastically eliminated vertices in
SVE. This approach exhibits the trade-offs of both SVE and FVE-GU. Similar to SVE,
this approach can re-attempt vertices at an earlier stage with the risk of duplication.
It also attempts to serve larger cliques, assuming the reception of the stochastically
eliminated vertices, at the risk of generating non-instantly decodable packets for the
receivers inducing these vertices. It is worth testing whether such a combination could
result in a better completion delay.

7.5 Proposed Algorithm

Given the vertex and graph update approaches described in Section 7.4, we can apply the
same algorithms of the continuous feedback problem, described in Section 5.4, to generate
efficient coded transmission in the intermittent feedback problem. In our description of
the algorithm operation, we define virtual states representing the graph transitions after
using a vertex and graph update approach within the feedback period.

In each visited virtual state, either the maximum weighted clique selection or the maximum weight vertex search algorithms is executed, and the packets of the selected clique is transmitted. If the process is still within the feedback period after this transmission, the vertices of this clique are eliminated from the graph according to one of the four approaches described above. The vertex adjacency in the resulting graph can also be updated if FVE-GU or SVE-GU are employed. Once this new graph is estimated, the subsequent maximum weight clique is selected and so on. This will sequentially generate $T_f$ cliques for the transmissions within the feedback periods. Now, when the actual feedback arrives at the end of the feedback period, the sender makes the correct adjustments to the graph, according to the received feedback, to reach a new actual state. Afterwards, the whole process is re-executed along the new feedback period. In case there is no remaining vertices in the graph while the process is still within a feedback period, the sender keeps retransmits the previously generated cliques, after the previous feedback instant, in the remaining transmissions until the arrival of the new feedback. This process is re-executed until the sender receives a completion acknowledgement from all receivers.

7.6 Simulation Results

In this section, we compare through extensive simulations the performances of the four vertex and graph update approaches with both the maximum weighted clique selection algorithm (denoted by “opt”) and the maximum weight vertex search algorithm (denoted by “srh”). We also compare the performances of these approaches to those of the continuous feedback (CF) IDNC algorithms, as a performance benchmark for IDNC. For all the above cases, we will employ the $L_3$ norm realization of our proposed algorithms in Section 5.4. We will also use the simulation settings employed in Section 5.5.
Figures 7.1 and 7.2 depict the comparison of the average completion delay and average completion feedback delay, respectively, achieved by each of the different algorithms, for multicast and broadcast scenarios, against $T_f$, for $M = 60$ and $N = 30$. Figures 7.3 and 7.4 depict the same comparisons against $M$, for $N = 30$ and $T_f = 5$. Figures 7.5 and 7.6 depict the same comparisons against $N$ for $M = 60$ and $T_f = 5$. Finally, Figure 7.7 depicts the same comparisons against $\mu$ for $M = 60$, $N = 30$ and $T_f = 5$. For all the figures, the packet erasure probabilities change from frame to frame in the range $[0.01,0.3]$, while maintaining $p = 0.15$.

From all the figures, we can see that the FVE approach achieves the best completion delay and best completion feedback delay, compared to the other three approaches, for both the optimal and search clique selection algorithms. This clearly shows that the different trade-offs introduced in the other three approaches tend to degrade the performance rather than improving it.

Another important observation is the degradation in average completion delay obtained in the intermittent feedback scenario, compared to the continuous feedback scenario, which naturally increases with the increase of the feedback period. However, for a relatively large network setting ($M = 100$, $N = 30$), a considerable feedback period value ($T_f = 5$) and a broadcast setting, this degradation reaches 3 and 3.5 extra transmissions for the FVEopt and FVEsrh algorithms, respectively, compared to their corresponding continuous feedback algorithms. This results in a maximum degradation in the frame delivery duration (from the start of the frame transmission until its reception at all receivers) of 6% and 6.5% for FVEopt and FVEsrh, respectively. These values are clearly tolerable given the 80% reduction in the feedback frequency, a reduction that is of extreme importance in many practical network settings as explained in the beginning of the chapter.
Chapter 7. IDNC with Intermittent Feedback

Figure 7.1: Average completion delay vs $T_f$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 7.2: Average completion feedback delay vs $T_f$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 7.3: Average completion delay vs $M$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 7.4: Average completion feedback delay vs $M$
Chapter 7. IDNC with Intermittent Feedback

Figure 7.5: Average completion delay vs $N$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Chapter 7. IDNC with Intermittent Feedback

Figure 7.6: Average completion feedback delay vs $N$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 7.7: Average completion and feedback delays vs $\mu$
7.7 Non-Periodic and Asynchronous Intermittent Feedback

In different network settings and applications, the intermittent feedback may not be periodic nor synchronous. In window based flow controlled applications, the feedback received from the receivers may come in variable intervals, depending on the network congestion. These non-periodic intervals may be also of random duration in some other applications. In the roadside to vehicle network example, the sender may require to receiver the acknowledgements of different vehicles at different times to reduce their signalling interference. Despite the fact that our SSP formulation in Section 7.2 cannot model most of the above scenarios, all of these scenarios have the same underlying geometric structure and evolution properties. Consequently, the proposed algorithms can still efficiently minimize the IDNC completion delay for these conditions but just need to be adjusted to fit them.

For the case of random intermittent feedback intervals, the algorithms extend directly by performing the vertex and graph update operations until the feedback is received at any time. In this case, the completion delay, achieved by our algorithms in these scenarios, will be a sort of a weighted averaging of the values in Figures 7.1, whose value depends on the statistics of the feedback interval random variable.

For the case of asynchronous intermittent feedback, the algorithms can be adjusted to make the graph corrections for each receiver when their own feedback arrives. In other words, there will not be any specific global feedback instant, but rather the vertex/graph update and graph correction procedures will run on the vertices of each receivers separately, according to its own feedback instants. In this case, the performance of the algorithm will not change drastically as long as the same period is maintained for all the receivers. However, it may be affected by the time gaps between the feedback instants of the different receivers.
7.8 Conclusion

In this chapter, we considered the problem of minimizing the completion delay of IDNC with intermittent feedback. We first extended the SSP formulation of the continuous feedback problem to the periodic and synchronous intermittent feedback problem, and showed it is more complicated to solve than the original one. We then showed that the intermittent feedback formulation has the same structure as the original formulation. We thus proposed the use of a similar approach to that used for continuous feedback, after making vertex elimination and graph update decisions for the un-acknowledged transmissions. Simulation results showed that the full vertex elimination with no graph update approach achieves the best performance compared to the other elimination and update approaches. This dominant approach also achieves a tolerable degradation against the continuous feedback performance, while using a much lower feedback frequency. We finally discussed the modifications needed to apply our proposed solutions in cases of non-periodic and asynchronous intermittent feedback conditions.
Chapter 8

IDNC with Lossy Feedback

8.1 Motivation

In several network settings, such as cellular and WiMAX systems, a high level of protection for feedback packets can be employed and thus feedback packet loss events will be very unlikely to occur. This lossless feedback scenario allows the direct use of our proposed algorithms, in Chapter 5, to efficiently reduce the IDNC completion delay in these network settings. However, several other network settings cannot guarantee the correct arrival of each feedback packet at the sender. One example can be found in sensor networks, in which the feedback packets sent from a sensor can be lost due to their low transmission power and/or the possible interference with the feedback of other sensors.

In these lossy feedback scenarios, the sender will receive feedback packets from only a subset of the receivers after a given transmission and thus the status of these receivers can be updated in the IDNC graph. For the other receivers, the latest status of packet reception and requests will be unknown. Consequently, the sender must blindly estimate the status of these receivers, in order to perform the subsequent IDNC transmission. In this following transmission, the sender may receive feedback packets from some of these...
receivers but will lose them for others. Consequently, the sender must continuously perform partially blind IDNC decisions until a correct completion feedback is received from all the receivers.

In this chapter, we will study the problem of completion delay minimization for IDNC with lossy feedback. We will start by formulating this problem as a partially observable stochastic shortest path problem and then employ this formulation to design efficient algorithms. For fair comparison with the original accurate feedback formulation and algorithms in Chapter 5, in terms of feedback frequency, we still assume that a receiver does not send any feedback unless it is targeted by a packet. In other words, if a feedback is lost by one of the targeted receivers, the sender will not get any feedback from this receiver until its next successfully received transmission in which it is targeted. We also assume that each feedback sent from a receiver includes acknowledgements of all previously received packets. Finally, we assume channel reciprocity, which means that the packet erasure probabilities seen by any receiver $i$, on both forward (sender to receiver) and reverse (receiver to sender) links, are the same and are both equal to $p_i$.

### 8.2 Problem Formulation

The minimum expected completion delay problem for IDNC with lossy feedback can be formulated as a PO-SSP problem as follows. The state space $S$, action spaces $A(s)$, state-action transition probabilities $P_a(s, s')$ and state action costs $c(s, a)$ for this PO-SSP are exactly defined as their corresponding entities in the original accurate feedback SSP formulation, illustrated in Sections 5.2.1, 5.2.2, 5.2.3 and 5.2.4, respectively. To complete the specification of the PO-SSP, we need to define the two following tuples.
8.2.1 Observation Spaces $\mathcal{O}(\kappa)$

Since any action represents a selection of a clique $\kappa$ for a given IDNC transmission, any observation in response to this action will represent the fact of receiving a feedback or not from the set of targeted receivers. An observation $o \in \mathcal{O}(\kappa)$ is a vector of length $M$, in which the entry $o_i = 1$ iff both $i \in T(\kappa)$ and the sender did not hear a feedback from $i$, and is equal to zero otherwise.

8.2.2 Conditional observation probabilities $Q_\kappa(s', o)$

To derive an expression for $Q_\kappa(s', o)$, we first define the set $\mathcal{P}(s', \kappa)$ of parent states of $s'$ having $\kappa$ in their action spaces. This set can be mathematically expressed as:

$$\mathcal{P}(s', \kappa) = \left\{ s | \kappa \in \mathcal{A}(s), P_\kappa(s, s') > 0 \right\} .$$  \hspace{1cm} (8.1)

We also define the probability $Q_\kappa(s, s', o)$ as the probability of observing observation $o$ when the process is at state $s'$, after selecting clique $\kappa$ when the process was at state $s \in \mathcal{P}(s', \kappa)$. To compute this probability, we define three sets on the same lines of (5.1) and (5.2) as follows:

$$\mathcal{X}_F = \left\{ i \in T(\kappa(s)) \mid \varphi_i(s) > \varphi_i(s'), o_i = 0 \right\} \hspace{1cm} (8.2)$$

$$\mathcal{X}_{NF} = \left\{ i \in T(\kappa(s)) \mid \varphi_i(s) > \varphi_i(s'), o_i = 1 \right\} \hspace{1cm} (8.3)$$

$$\mathcal{Y} = \left\{ i \in T(\kappa(s)) \mid \varphi_i(s) = \varphi_i(s') \right\} . \hspace{1cm} (8.4)$$

The first set includes the targeted receivers that have successfully received the IDNC packet generated from $\kappa(s)$ and their feedback was received. The second set includes the receivers that have successfully received the packet but their issued feedback was not received by the sender. The third set includes the targeted receivers that have lost the IDNC packet. In this case, there will be no splitting of the set as a receiver not receiving
a packet does not generate a feedback. Based on the definitions of these sets, \( Q_\kappa(s, s', o) \) can be expressed as follows:

\[
Q_\kappa(s, s', o) = \prod_{i \in \mathcal{X}_P} q_i^2 \cdot \prod_{i \in \mathcal{X}_{N_P}} q_i p_i \cdot \prod_{i \in \mathcal{Y}} p_i .
\] (8.5)

Finally, we can express the conditional observation probability as:

\[
Q_\kappa(s', o) = \sum_{s \in \mathcal{P}(s, a)} Q_\kappa(s, s', o) .
\] (8.6)

### 8.2.3 Belief States and Belief Updates

To identify the different belief states, we first need to quantify the probabilities of the different uncertainty sources in the process. In the above PO-SSP formulation, an uncertainty in the exact state of the process occurs in case the sender does not receive a feedback from one or more targeted receivers. This no-feedback-reception event from a targeted receiver \( i \) could mean one of two sub-events:

1. The packet was not received by \( i \) and thus it did not issue a feedback. This event can occur with probability \( p_i \).

2. The packet was received by \( i \), and \( i \) issued a feedback packet that did not arrive at the sender. This event can occur with probability \( q_i p_i \).

If any of these two sub-events occurs, and if the packet intended for receiver \( i \) is packet \( j \), then the position \( f_{ij} \) in the feedback matrix will be uncertain. It can be equal to 1 (packet not received) if the first event occurred or 0 (packet received) if the second event occurred. Consequently, the no-feedback event from the targeted receiver \( i \) with packet
Chapter 8. IDNC with Lossy Feedback

\( j \), will render \( f_{ij} \) a Bernoulli random variable defined as:

\[
P(f_{ij} = 1 | a_i = 1) = \frac{p_i}{p_i + q_i p_i} \quad (8.7)
\]

\[
P(f_{ij} = 0 | a_i = 1) = \frac{q_i p_i}{p_i + q_i p_i} . \quad (8.8)
\]

All the receivers for which a feedback is received do not create any uncertainty and thus will not contribute to the belief state definition. Due to our assumption that a targeted receiver sends acknowledgements of all received packets, a received feedback from this receiver removes all previous uncertainties in its \( f_{ij} \) entries. Finally note that once an entry moves from being uncertain to being certain, it remains certain for the rest of the process until absorption. Indeed, an acknowledged packet at a given time cannot be suspected of not being received at later times.

Define the uncertainty matrix \( F_u \), in which all acknowledged and un-attempted entries are equal to 0, 1 and \(-1\), according to \((3.1)\), and all uncertain entries due to no-feedback events are set to \( x \). Having this uncertainty feedback matrix defined, we can now define the belief space \( B \) as follows. For each possible uncertainty matrix \( F_u \), that can occur from the beginning of the frame transmission until its completion, we generate a belief state \( b(F_u) = [b(s|F_u)] \forall s \in S \), whose entry \( b(s|F_u) \) is the probability of being in state \( s \) given the uncertainty matrix \( F_u \). The states with non-zero values in \( b(s|F_u) \) are those representing all possible combinations of replacing each \( x \) entry of \( F_u \) by 0 (assuming it is received) or 1 (-1) (assuming it is (not) wanted and not received). Thus, the number of non-zero entries in \( b(F_u) \) is equal to \( 2^\varsigma \), where \( \varsigma \) is the number of \( x \) entries in \( F_u \). Defining \( \lambda_{i,1} \) and \( \lambda_{i,0} \) as the number of \( x \) entries, in the \( i \)-th row, replaced by \( \{1,-1\} \) and 0, respectively, to reach the shape of \( F(s) \), we can thus express \( b(s|F_u) \) as:

\[
b(s|F_u) = \prod_{i \in M} \left( \frac{p_i}{p_i + q_i p_i} \right)^{\lambda_{i,1}} \left( \frac{q_i p_i}{p_i + q_i p_i} \right)^{\lambda_{i,0}} . \quad (8.9)
\]
The entries of $F^u$ are updated after every transmission and observation of the feedback. If feedback is received from receiver $i$, each $x$ entry in the $i$-th row of $F^u$ are set to 0,1 or -1 depending on the feedback information of this receiver. Any no-feedback events for a targeted receiver $i$, between two feedback events, result in $x$ values at their corresponding packet locations in the $i$-th row. After updating the uncertainty matrix, we can update the belief state of the process according to (8.9). Note that the process is still Markovian since the future belief depends only on the current uncertainty matrix, current action and current observation.

Using the above parameters, we can derive the belief transition probabilities from (2.5) and (2.6). Finally, all belief-action costs will remain one since setting all state-action costs to one makes (2.7) a summation of a probability distribution over all its sample space, which is equal to 1.

### 8.3 Formulation Properties and Proposed Algorithms

According to the above PO-SSP formulation, one of its major properties is its operation as an overlay from the sender’s perspective, while the actual process running beneath have the same properties as the accurate feedback problem. Consequently, the extremely large dimensions of the state and action spaces in the underlying SSP makes the dimensions of the belief space in the PO-SSP even larger. Due to this extreme sizes of the different PO-SSP components, solving this PO-SSP optimally is impossible.

Nonetheless, this property also means that the underlying process have the same geometric structure and graph evolution properties as the accurate feedback problem. Consequently, this underlying process can be efficiently optimized using the same algorithms designed in Section 5.4. In other words, employing the WoRLT strategy in each transmission can both brings the process closest to absorption and maximizes the coding density in the IDNC graph.
Similar to Chapter 7, the only difficulty to directly implement this approach in the lossy feedback problem is that the sender should perform network coding decisions without knowing the outcome of the attempted and un-acknowledged transmissions due to no-feedback events. To be capable of applying the efficient accurate feedback algorithms in each transmission of the lossy feedback problem, the sender needs to make blind vertex and graph update decisions, after each transmission, for the non-acknowledged vertices of the served clique. To do so, we propose to employ one of the four vertex and graph update approaches, compared for the intermittent feedback case in Chapter 7.

In the lossy feedback context, we can see that the FVE approach results in an IDNC graph that is a subgraph of all the states having non-zero entries in the belief state. Consequently, using FVE guarantees innovation in all generated packets regardless of the uncertainty in the current partially observed state.

On the other hand, SVE would better represent the reception probabilities of the receivers, compared to FVE, and thus can re-attempt the non-received vertices in an earlier stage, compared to FVE. Although SVE falls in the risk of re-attempting received vertices, which was shown to achieve worse performance in the intermittent feedback case, this early re-attempting of vertices results in an earlier update of the feedback matrix (since only targeted receivers can send back feedback), especially towards the end of the recovery phase. This may be of greater importance in this lossy feedback context than guaranteeing packets’ innovation for all targeted receivers. Indeed, earlier updates of the graph, especially at the end of the recovery phase, can help the sender make better decisions, especially when these updates are polled according to the channel quality as in SVE (more polls for lower quality channels). This may result in a lower reception completion and completion feedback delays. In the following section, we will test the importance of these two factors in achieving a better performance in the lossy feedback scenarios.
8.4 Simulation Results

In this section, we compare through extensive simulations the performances of the four vertex and graph update approaches, using both the maximum weight clique selection algorithm (denoted by “opt”) and the maximum weight vertex search algorithm (denoted by “srh”). We also compare the performances of these approaches to those of the accurate feedback (AF) IDNC algorithms, as a performance benchmark for IDNC. For all the above cases, we will employ the $L_3$ norm realization of our proposed algorithms in Section 5.4. We also use the simulation settings employed in Sections 5.5 and 7.6. The packet erasure probabilities of all receivers change from frame to frame in the range $[0.01,0.3]$, while maintaining $p = 0.15$.

Figure 8.1 depicts the comparison of the average completion delays and average completion feedback delay, respectively, achieved by the different algorithms, for both multicast and broadcast scenarios, against $\mu$, for $M = 60$ and $N = 30$. Figures 8.2 and 8.3 depict the same comparisons against $M$ for $N = 30$. Finally, Figures 8.4 and 8.5 depict the same comparisons against $N$ for $M = 60$.

From Figure 8.1, we can see that FVE slightly outperforms SVE at low demand ratios where as SVE slightly outperforms FVE at high demand ratios. This result is also shown in all other figures, where FVE can outperform SVE in some points, when $\mu = 0.5$ but never when $\mu = 1$.

This can be explained in the light of the characteristics of the two approaches as follows. At low demand ratios, the number of vertices in the IDNC primary graph are relatively smaller compared to that at high demand ratios. Consequently, the time needed by the sender to attempt all these vertices, using FVE, is small. We know, from Section 7.4.1, that FVE re-attempts unacknowledged vertices when all the vertices of the graph are once attempted. Consequently, if the last vertex or several vertices of any receiver are attempted but unacknowledged, they will wait for a short time before being re-attempted, which helps both the sender in rapidly knowing the actual status
Figure 8.1: Average completion and feedback delays vs $\mu$
Figure 8.2: Average completion delay vs $M$
Figure 8.3: Average completion feedback delay vs $M$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 8.4: Average completion delay vs $N$

(a) Multicast scenario, $\mu = 0.5$

(b) Broadcast scenario, $\mu = 1$
Figure 8.5: Average completion feedback delay vs $N$
of these vertices and the receiver to rapidly recover its missing packet(s). Note that we only consider the last vertex or vertices of a receiver, attempted after the last received feedback from this receiver, since any unacknowledged vertices before this feedback have already been acknowledged by this feedback.

At high demand ratios, the larger size of the IDNC primary graph makes the time for FVE to attempt all vertices longer. Consequently, all receivers, whose last vertex or several vertices were attempted but unacknowledged, will have to wait longer for FVE to re-attempt them. This effect increases for the receivers with the smaller Wants sets. This may cause a longer delay for the sender to re-attempt these packets. On the other hand, SVE reduces this effect since it stochastically leaves the attempted vertices in the graph, which give them a chance to speed up their transmission re-attempt, their recovery and/or their feedback.

From the other figures, we can see that SVE dominates the performance in most values of $M$ and $N$ with some few exceptions at low demand ratio. As $M$ and $N$ increase, the effect of early re-attempts offered by SVE becomes more important in completing the frame transmission earlier. We can also see from all the figures that employing graph updates with both FVE and SVE results in worse performance for all metrics.

Finally, we can observe a degradation in the average completion delay obtained in the lossy feedback scenario compared to the accurate feedback scenario. However, for a relatively large network setting ($M = 100$, $N = 30$), a worst erasure probability of 0.3, and a broadcast setting, this degradation reaches 4.7 and 5.6 extra transmissions for the SVEopt and SVEsrh algorithms, respectively, compared to their corresponding continuous feedback algorithms. This results in a maximum degradation in the frame delivery duration (from the start of the frame transmission until its reception at all receivers) of 9% and 10.5% for SVEopt and SVEsrh, respectively, compared to the accurate feedback algorithm performance. These values are clearly tolerable in such very large network and up to 30% loss rate of feedback, which is typically very high for signalling information.
They are also tolerable in very unorganized and fast changing network environments such as roadside to vehicle networks.

8.5 Conclusion

In this chapter, we considered the problem of minimizing the completion delay of IDNC with lossy feedback. We first modeled this problem as a PO-SSP problem, having the underlying SSP process operating similar to that of the accurate feedback problem. Although the designed PO-SSP inherits the extremely large size and solution intractability, from the original SSP formulation, it also inherits its structure and properties in the underlying SSP level. However, the overlay process lacks some knowledge of the actual underlying process evolution. Consequently, we proposed the use of the efficient algorithms designed in Section 5.4, after making vertex and graph update decisions for the un-acknowledged transmissions. Simulation results showed that the stochastic vertex elimination approach achieves the best performance for most parameter ranges, especially at high demand ratios. At low demand ratios, the full vertex elimination approach outperforms the stochastic one in some small parameter ranges. Simulations also show that the proposed algorithms achieve a tolerable degradation against the accurate feedback performance for considerably high feedback loss rates.
Chapter 9

Conclusion and Future Directions

9.1 Thesis Conclusion

In this thesis, we aimed to minimize the effect of the IDNC drawbacks, with respect to FNC, in order to enjoy all its desirable benefits, which makes it attractive over FNC, without suffering much from its weaknesses. These drawbacks of IDNC with respect to FNC can be summarized in two main problems. The first problem is its inability to guarantee the decoding of a new packet at each receiver in each transmission, which may severely affect its completion delay. The second problem is its need for full feedback in order to operate properly.

To achieve the target of this thesis, we addressed the following questions:

1. What is the receiver and packet selection strategies that can maximize the expected coding density in each transmission and over the transmission horizon of a frame of packets?

2. Given the knowledge of received and lost packets at different receivers and their packet erasure probabilities, how can one minimize the expected completion delay of a frame of packets in IDNC for both multicast and broadcast scenarios?
3. How can we adaptively select between IDNC and FNC the scheme that is expected to achieve the smaller completion delay?

4. How can we achieve the minimum completion delay in IDNC under intermittent and lossy feedback scenarios?

For these questions, here is a summary of our findings and conclusions:

### 9.1.1 Densification of Coding Opportunities in IDNC

The following list includes a summary of our conclusions from the study of IDNC graph evolution.

- Serving packets requested by a larger number of receivers maximizes the coding density in the beginning of the recovery transmission phase but fails in maintaining this maximization due to the lack of packets satisfying this property.

- Targeting the maximum number of receivers, having the largest numbers of missing packets and erasure probabilities in a layered fashion (which we called the WoRLT strategy), maximizes the expected number of coding opportunities and achieves a monotonic increase in the coding density. It thus maximizes the coding density not only for one step, but for all future steps.

- Vertices in the maximum cliques of the IDNC graph are mostly induced by the receivers having the smallest number of requested packets and erasure probabilities. Consequently, targeting the largest number of such receivers in each transmission does not maximize the coding density of the graph.

### 9.1.2 Completion Delay Minimization in IDNC

The following list includes a summary of our conclusions from the study of the completion delay minimization in IDNC.
Chapter 9. Conclusion and Future Directions

- Applying the WoRLT strategy in every IDNC transmission brings the process closer to completion and maximizes the expected coding density of the subsequent IDNC graphs. This strategy is thus efficient in minimizing the expected completion delay for IDNC.

- This WoRLT strategy can be implemented using a maximum weight clique search over the IDNC graph. It can be also implemented using a quadratic time heuristic, which is more suitable for real-time applications.

- The proposed algorithms considerably outperform the maximum clique algorithms, serving the maximum number of receivers and vertices in each transmission. The reason for this worse performance results from the effect of graph evolution. In other words, despite its targeting of the largest number of receivers in the very beginning of the recovery transmission phase, this policy results in very poor coding opportunities and density in the subsequent IDNC graphs, which greatly reduces its advantage in these subsequent transmissions.

- The proposed policy achieves a near-optimal completion delay performance in the broadcast scenario, with a maximum degradation of 1.9% in the frame delivery duration, compared to FNC.

- The proposed heuristic algorithms achieves a very close performance to the optimal maximum weight clique algorithms, and a maximum degradation of 4.4% in the frame delivery duration, compared to FNC in broadcast scenarios.

9.1.3 Adaptive IDNC-FNC Selection

The following list includes a summary of our conclusions from the study of adaptive selection between IDNC and FNC to minimize the overall completion delay.

- The performance superiority of either IDNC and FNC in different ranges of number
of receivers and demand ratios, can be determined by comparing cleverly designed metrics for IDNC, using random graph theory, to corresponding FNC parameters.

- For all designed metrics, the optimal selection between IDNC and FNC is almost achieved with a maximum degradation between 0.5 to 1 transmission.

- This near-optimal selection can be achieved without adding extra performance fluctuation, without considering loss patterns and with a low parameter computation rate.

### 9.1.4 Completion Delay Minimization with Intermittent Feedback

The following list includes a summary of our conclusions from the study of completion delay minimization in IDNC with intermittent feedback.

- The formulation of the optimal completion delay problem for IDNC with intermittent feedback has more complicated action possibilities than that of the continuous feedback problem, but has its same structure and properties.

- The proposed policy for continuous feedback can be employed in intermittent feedback scenarios, after making vertex and graph update decisions for the attempted and unacknowledged vertices.

- The approach that eliminates all attempted and unacknowledged vertices with no graph updates achieves the best performance compared to other elimination and update approaches.

- Our proposed algorithms with the above vertex and graph update approach achieve a maximum degradation in the frame delivery duration of around 6%, compared to the continuous feedback performance, in large networks and while using only 20% of the continuous feedback frequency.


9.1.5 Completion Delay Minimization with Lossy Feedback

The following list includes a summary of our conclusions from the study of completion delay minimization in IDNC with lossy feedback.

- The partially observable formulation of the optimal completion delay problem for IDNC with lossy feedback is more complicated than that of the accurate feedback problem, but its underlying process has its same structure and properties.

- The proposed policy for continuous feedback can be employed in the lossy feedback scenarios after making vertex and graph update decision for the attempted and unacknowledged vertices.

- The approach that stochastically eliminates attempted and un-acknowledged vertices, according to the reception success probabilities of their inducing receivers, achieves the best performance for most parameter ranges, especially at high demand ratios.

- At low demand ratios, the approach eliminating all attempted and unacknowledged vertices achieves the best performance in some small ranges of the different parameters.

- Our proposed algorithms with stochastic vertex elimination achieve a maximum degradation in the frame delivery duration of around 10%, in large networks and a feedback loss rate of 30%.

From all these conclusions, we believe we have addressed the major concerns against the practical implementation of IDNC, and presented reasonable and applicable solutions for it. Given its strength points over FNC and the proposed solution in this thesis, we thus recommend the selection of IDNC as a revolutionary recovery transmission approach in future wireless networks, and foresee it as an important component in future wireless communication standards.
9.2 Future Directions

The work in this thesis can be extended in different directions, which can be classified into three categories, namely, the network topology, channel model and transmission mode. We will detail the potential extensions in each of these categories in the following sections.

9.2.1 Network Topology

In this thesis, we only considered the point-to-multipoint network topology, where a single sender is broadcasting or multicasting information to multiple receivers. Nonetheless, we can study the same problems of this thesis in other more complicated network topologies, which may raise even more interesting problems.

Relay-Assisted Cellular Networks

One network topology direction is the relay-assisted cellular networks. In these networks, the sender is assisted with multiple relays, distributed in the coverage area of the sender, with the aim of extending its coverage. A well-known example of such topology is the 802.16j standard. In this topology, the sender broadcasts the frame of packets once to all relays and receivers, and each of them receives different subsets of the frame packets. Given this setting, several interesting problems can be studied:

- Completion and decoding delay minimization: In this problems, the proposed solution should not only select the IDNC packets to be encoded in each transmission but should also select the transmitter (either the main sender or any of the relays) that should be used in each transmission in order to optimize the desired delay parameter.

- Joint power and completion delay minimization: If different relays have different power abilities, we can consider the problem of selecting the transmitter and IDNC
packet, in each transmission, with the purpose of jointly optimizing the completion delay and the overall power consumption.

- Joint power and decoding delay minimization: This is the same problem as the previous one but with decoding delay replacing the completion delay in the joint optimization problem.

- Distributed selection of transmitters and IDNC packets: In this problem, the decision about the transmitter and IDNC packet selection for each transmission should be carried on in a distributed fashion.

**MANETs and VANETs**

Another interesting network topology is the mobile ad-hoc networks and the vehicular ad-hoc networks. In such topologies, different nodes (or vehicles) have different information packets that need to be sent to a subset or all the other nodes. These nodes may or may not be in the coverage areas of each other. When each of these nodes transmits its packets once, different nodes will acquire different subsets of the transmitted packets, whether they want these packets or not. In this case, we can employ IDNC in the subsequent transmissions in order to relay all the packets to their intended destinations. In this case, the node that should transmit in each transmission and the packet combination it should send can be optimized in order to minimize the completion or decoding delay. Since there is no central processing unit, the node and packet selection for each transmission should be computed efficiently in a distributed fashion, so that no or very limited collisions occur.

**9.2.2 Channel Model**

In this thesis, we considered an independent erasure channel model, in which packets can be independently received or lost from one transmission to the next. Although this
model is widely used in most related works, we can expand our study to more complicated channel models such as:

- Gilbert-Elliott channel model: This channel model represents Markovian transitions between Good and Bad channel states. Each receiver correctly receives all the packets when it is in the Good state, and misses all the packets when it is in the Bad state.

- Multilevel channel model. This channel model is also Markovian but transitions between several states, each of which allowing the receivers to receive with a different erasure probability or with a different transmission rate.

- Multichannel model: In this model, each receiver has different erasure probabilities or transmission rates on different parallel channels (like in OFDMA). Thus, the sender should select multiple coded packets to be transmitted on these multiple channels in order to minimize the completion or decoding delay.

In all the above models, we can study the completion or decoding delay optimization problems given a full or limited knowledge of the channel states at different receivers.

### 9.2.3 Transmission Mode

In this thesis, we only considered optimizing IDNC in a rateless transmission mode. In other words, the sender is allowed to transmit coded packets from the same frame of source packets until all receivers receive their requested packets. We can extend this work by considering other transmission modes such as:

- Restricted transmission mode: In this mode, a deadline is set for the transmission of each frame. In this scenario, we can optimize coded packets in order to maximize the average reception ratio (defined as the ratio of the number of received packets to the total number of packets) and success probability (defined as the probability of satisfying all packet requests before the deadline).
• Non-framed transmission mode: In this mode, the concept of frames of packets that needs to be transmitted before moving to new frames does not exist. Thus, newly arriving packets should be directly considered for transmission along with previously lost packets. Given the arrival of new packets in each time slot and the availability of previously lost packets in the sender’s buffers, the proper IDNC packet selection should be made in order to optimize different parameters, such as throughput and queuing time.
Bibliography

[1] Air interface for fixed and mobile broadband wireless access systems - amendment for physical and medium access control layers for combined fixed and mobile operation in licensed bands, May 2005.


